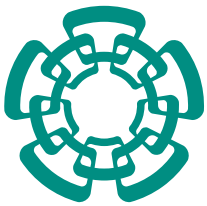


Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional

Unidad Zacatenco
Departamento de Computación

Aspectos de seguridad de Bitcoin y su aplicación en una alternativa de infraestructura de llave pública

Tesis que presenta

Abraham Jesús Basurto Becerra

para obtener el grado de

Maestro en Ciencias en Computación

Director de la tesis:

Dr. Francisco José Rambó Rodríguez Henríquez

México, D.F. **Diciembre, 2015**

«*Ipsa scientia potestas est*»
—Sir Francis Bacon

A mis padres

RESUMEN

Cuando se emplea criptografía de llave asimétrica en una comunicación es imprescindible el poder verificar que la llave pública de una entidad es auténtica. En caso de no ser esto posible se corre el riesgo de sufrir un ataque del intruso de en medio entre otras, comprometiendo de esta manera la seguridad de la comunicación.

En los albores de la criptografía de llave pública, en su artículo seminal de 1976, Whitfield Diffie y Martin Hellman propusieron un directorio público donde se colocarían las llaves de cifrado junto con el nombre y la dirección del propietario. Ellos consideraron que el problema del acceso a éste era fácil de resolver, ya que la lectura siempre estaría permitida y como la escritura sería ocasional, se podían emplear elaborados mecanismos de protección.

En 1978, Loren Kohnfelder propuso el concepto de certificados digitales. En ellos, una autoridad certificadora enlaza un nombre a una llave pública mediante una firma electrónica. La Unión Internacional de Telecomunicaciones adoptó los certificados y realizó el estándar X.509 en 1988, mismo que ha sido adoptado por la industria como el modelo a emplear en infraestructura de llave pública.

El objetivo principal del desarrollo de la infraestructura de llave pública es: hacer posible la obtención de llaves públicas de manera segura, conveniente y eficiente. La objeción principal a los sistemas de infraestructura de llave pública basados en X.509 es el empleo del modelo de confianza jerárquico. Se confía ciegamente en las autoridades certificadoras. En el caso de que alguna autoridad certificadora sea comprometida, el atacante puede emitir certificados falsos, suplantando a cualquier entidad que desee, sin generar sospecha alguna de parte de la víctima potencial de tal engaño.

Este mismo problema se presenta en la gestión de dinero electrónico confiable y seguro. En los modelos iniciales, como el presentado por David Chaum en 1983, se requería de una tercera entidad (normalmente un banco) la cual debía emitir el dinero electrónico a los usuarios, y en caso de que tal emisor fuera comprometido, el atacante sería capaz de acuñar dinero electrónico a placer.

En 2008, Satoshi Nakamoto publicó *Bitcoin: A Peer-to-Peer Electronic Cash System*, donde combina varios conceptos desarrollados por propuestas previas de sistemas de dinero electrónico, para lograr un sistema completamente descentralizado. La innovación clave de Bitcoin está en el uso de un sistema distribuido que emplea un algoritmo de «prueba de trabajo», en el que aproximadamente cada diez minutos se lleva a cabo una «elección», permitiendo a la red descentralizada llegar a un consenso global acerca del estado de las transacciones.

En Bitcoin no existe una entidad todopoderosa que gobierne la emisión de dinero. Esto se logra empleando una red punto a punto (P2P), en la cual, los participantes concursan para producir nuevas monedas, así como, por obtener una ganancia al procesar y asegurar las transacciones. Bitcoin ha operado desde 2009 sin haberse encontrado alguna vulnerabilidad que comprometa la seguridad del sistema y que, por lo tanto, permita a alguien el poder crear u obtener bitcoins sin seguir el protocolo y competir con los demás participantes.

Todas las transacciones realizadas desde el inicio de Bitcoin se encuentran almacenadas en el *blockchain* — la bitácora global de Bitcoin—. Cada cliente de la red de Bitcoin almacena de forma independiente una copia del *blockchain*. Su función es prevenir que algún individuo trate de gastar más de una vez una moneda electrónica así como evitar que alguien trate de modificar transacciones ya realizadas.

El objetivo de este trabajo es estudiar los conceptos que emplea Bitcoin para asegurar sus transacciones, para posteriormente poder aplicarlos al campo de los sistemas de infraestructura de llave pública, teniendo como finalidad eliminar la necesidad autoridades certificadoras raíz y así lograr un sistema que opere de forma descentralizada.

Se propone emplear un *blockchain* para realizar lo que plantearon hace casi 40 años Diffie y Hellman: un directorio público, con permiso de lectura para cualquiera y con un proceso de escritura realizado por una red punto a punto compitiendo por crear nuevos bloques. Aumentando con cada bloque la dificultad de poder modificar maliciosamente los datos.

Al eliminar la necesidad de confianza ciega en una autoridad certificadora se logra una total transparencia en la emisión de certificados raíz. También se diluye el riesgo de compromiso, ya que no existe una entidad central que emita los certificados.

ABSTRACT

When asymmetric key cryptography is used to secure a communication, being able to verify an entity's public key is essential. When this is not possible, there exists a high risk that an oponent may launch successful man-in-the-middle and misbinding-identity attacks, compromising in this way the communication security.

At the dawn of public key cryptography, Whitfield Diffie and Martin Hellman in their seminal paper of 1976 proposed a public directory where all the enciphering keys will be placed along with the name and address of the owner. They considered that the access to this directory will be an easy problem, being that reading access will always be granted, and at the same time the data will be infrequently modified. Hence, elaborated write protection mechanisms could be employed.

In 1978, Loren Kohnfelder proposed the concept of digital certificates, where a certificate authority binds a name with a public key by using a digital signature. Certificates were adopted by the International Telecommunication Union and employed in the X.509 standard released in 1988, which was embraced by the industry as the *de facto* model for public key infrastructure.

The main goal of the public key infrastructure development is: to make possible obtaining public keys in a secure, convenient and efficient fashion. The principal objection to X.509 based PKI systems is its reliance in the hierarchical trust model. Certification authorities are blindly trusted. In the event that one CA is compromised, the attacker would be able to issue fake certificates, effectively supplanting any raising entity and without any suspicion of the potential victim of the attack.

The same problem exists in the secure and reliable management of electronic money. In the initial models, like the one presented by David Chaum in 1983, a trusted third party (normaly a bank) was required to mint and then issue the electronic money to the the users. In the event that the issuer gets compromised, the attacker will be able to issue money at will.

In 2008, Satoshi Nakamoto published *Bitcoin: A Peer-to-Peer Electronic Cash System*, where he combines several concepts previously developed in electronic money systems to achieve a completely decentralized system. The key innovation in Bitcoin is the use of a distributed system that employs a proof-of-work algorithm, in which approximately every 10 minutes an *election* is performed, this is how the decentralized network reaches consensus about the transactions current state.

In Bitcoin, there is no almighty entity that rules the issuing of money. This is achieved by a peer-to-peer network, in which the participants compete to mint new coins and also to obtain a fee by processing and securing the transactions. Bitcoin has been operating since 2009 without any vulnerability being found that compromises the security of the system and by this allowing someone to issue or obtain bitcoins without following the protocol of competing with the other participants.

Every transaction made since the beginning of Bitcoin is stored in the *blockchain* —Bitcoin's global ledger—. Each client in the Bitcoin network stores an independent copy of the *blockchain*. Its purpose is to prevent double spending and transaction manipulation.

The objective of this work is to apply the concepts that Bitcoin employs to secure its transactions to the field of the public key infrastructure systems, having as a goal getting ridden of the need of root certification authorities and by this enabling a decentralized system.

Using a *blockchain* to achieve what Diffie and Hellman stated almost 40 years ago, a public directory with read access to anyone and a writing process performed by a peer-to-peer network competing to create new blocks is proposed. Increasing with every new block the difficulty of malicious data modifications.

By getting ridden of the blind trust in a certification authority, total transparency is achieved in the emission of root certificates. Also the risk of being compromised is disseminated as there does not exist a central certificate issuer.

AGRADECIMIENTOS

«El agradecimiento es la memoria del corazón»

—LAO-TSÉ

Agradezco al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo económico brindado a lo largo de los dos años del programa de maestría.

Al proyecto SEP-CONACYT 180421, por el apoyo económico brindado para la obtención del grado.

Gracias Papá por enseñarme a nunca rendirme ante los problemas, gracias Mamá por enseñarme que el amor es la fuerza más grande que existe. Agradezco a Dios el haberme dado unos padres tan amorosos. Siempre los llevaré en mi corazón.

Gracias a mi hermano, que ha sido un gran cómplice a lo largo de esta aventura llamada vida.

Agradezco de forma especial al Dr. Francisco Rodríguez Henríquez —mi asesor— por sus enseñanzas, su confianza, su paciencia, su orientación y su ayuda.

A mis revisores, la Dra. Sandra Díaz Santiago y el Dr. Luis Gerardo de la Fraga, por enriquecer mi trabajo de tesis con sus contribuciones.

A todos los excelentes investigadores del Departamento de Computación, los cuales me transmitieron un poco de su conocimiento a lo largo de sus clases.

A todos mis compañeros con los cuales recorrí este camino, gracias por esos ratos amenos y desveladas que vivimos; por su amistad, su apoyo, su confianza, sus consejos; por compartir su conocimiento y por hacer de mí una mejor persona.

A todo el personal administrativo y de auxiliares de investigación del Departamento de Computación. A Sofi, Feli y Erika por su amabilidad y por protegernos de los filosos dientes de la burocracia. Al Dr. Santiago, al Mtro. José Luis y al Ing. Arcadio por procurar que todos los servicios que usamos día a día para el desempeño de nuestras actividades se encuentren en óptimas condiciones.

Al Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional (Cinvestav) por permitirme ser parte de esta gran institución.

A. J. B. B.

Índice general

ACRÓNIMOS

ANSI	*American National Standards Institute*
	Instituto de Estándares Nacionales de los Estados Unidos
ARL	*Authority Revocation List*
	Lista de Revocación de Autoridades
ASIC	*Application Specific Integrated Circuit*
	Circuito Integrado de Aplicación Específica
ASN	*Abstract Syntax Notation*
	Notación Sintáctica Abstracta
BIP	*Bitcoin Improvement Proposal*
	Propuesta de Mejora de Bitcoin
CA	*Certificate Authority*
	Autoridad Certificadora
ccTLD	*Country Code Top Level Domain*
	Dominio de Nivel Superior de Código de País
CN	*Common Name*
	Nombre Común

CPU	*Central Processing Unit*
	Unidad Central de Procesamiento
CRL	*Certificate Revocation List*
	Lista de Revocación de Certificados
DN	*Distinguished Names*
	Nombre Distinguido
DNS	*Domain Name System*
	Sistema de Nombre de Dominio
DSA	*Digital Signature Algorithm*
	Algoritmo de Firma Digital
DSS	*Digital Signature Standard*
	Estándar de Firma Digital
ECDH	*Elliptic Curve Diffie Hellman*
	Diffie Hellman de Curva Elíptica
ECDSA	*Elliptic Curve Digital Signature Algorithm*
	Algoritmo de Firma Digital de Curva Elíptica
ECIES	*Elliptic Curve Integrated Encryption Scheme*
	Esquema Integrado de Cifrado de Curva Elíptica
ECMQV	*Elliptic Curve Menezes-Qu-Vanstone*
	Menezes-Qu-Vanstone de Curva Elíptica
FIPS	*Federal Information Processing Standard*
	Estándar Federal de Procesamiento de la Información
FPGA	*Field Programmable Gate Array*
	Arreglo de Compuertas Programables en Campo
GPU	*Graphics Processing Unit*
	Unidad de Procesamiento Gráfico
gTLD	*Generic Top Level Domain*
	Dominio de Nivel Superior Genérico
HTTPS	*Hypertext Transfer Protocol Secure*
	Protocolo Seguro de Transferencia de Hipertexto
ICANN	*Internet Corporation for Assigned Names and Numbers*
	Corporación de Internet para la Asignación de Nombres y Números
IDN	*Internationalized Domain Names*
	Nombre de Dominio Internacionalizado
IEEE	*Institute of Electrical and Electronics Engineers*

	Instituto de Ingenieros Eléctricos y Electrónicos
IETF	*Internet Engineering Task Force*
	Grupo de Trabajo de Ingeniería de Internet
ISO	*International Organization for Standardization*
	Organización Internacional de Estandarización
ITU	*International Telecommunication Union*
	Unión Internacional de Telecomunicaciones
JSON	*JavaScript Object Notation*
	Notación de Objetos de JavaScript
KEM	*Key Encapsulation Mechanism*
	Mecanismo de Encapsulamiento de Llaves
LFPIORPI	Ley Federal Para la Prevención e Identificación de Operaciones con Recursos de Procedencia Ilícita
NFC	*Near Field Communication*
	Comunicación de Campo Cercano
NIC	*Network Information Center*
	Centro de Información de Red
NIST	*National Institute of Standards & Technology*
	Instituto Nacional de Estándares y Tecnología de los Estados Unidos
O	*Organization*
	Organización
OCSP	*Online Certificate Status Protocol*
	Protocolo en Línea de Estado de Certificados
OSI	*Open System Interconnection*
	Interconexión de Sistemas Abiertos
OU	*Organizational Unit*
	Unidad Organizacional
P2P	*Peer to Peer*
	Punto a Punto
P2PKH	*Pay-to-Public-Key-Hash*
	Pago a Picadillo de Llave Pública
P2SH	*Pay-to-Script-Hash*
	Pago a Picadillo de Script
PGP	*Pretty Good Privacy*
	Privacidad Bastante Buena

PIB	Producto Interno Bruto
PKI	*Public Key Infrastructure*
	Infraestructura de Llave Pública
QR	*Quick Response*
	Respuesta Rápida
RA	*Registration Authority*
	Autoridad Registradora
RDN	*Relative Distringuished Name*
	Nombre Distinguido Relativo
RFC	*Request for Comments*
	Solicitud de Comentarios
RIPEMD	*RACE Integrity Primitives Evaluation Message Digest*
	Digesto de Mensaje de Evaluación de Primitivas de Integridad RACE
RR	*Resource Record*
	Registro de Recurso
S/MIME	*Secure / Multipurpose Internet Mail Extensions*
	Seguro / Extensiones Multipropósito de Correo de Internet
SECG	*Standards for Efficient Cryptography Group*
	Grupo de Estándares para Criptografía Eficiente
SHA	*Secure Hash Algorithm*
	Algoritmo de Picadillo Seguro
SLD	*Second Level Domain*
	Dominio de Segundo Nivel
SO	Sistema Operativo
SPV	*Simplified Payment Verification*
	Verificación Simplificada de Pagos
SSL	*Secure Sockets Layer*
	Capa de Conexión Segura
sTLD	*Sponsored Top Level Domain*
	Dominio de Nivel Superior Patrocinado
TLD	*Top Level Domain*
	Dominio de Nivel Superior
TLS	*Transport Layer Security*
	Seguridad de la Capa de Transporte
TTL	*Time to Live*

	Timpo de Vida
TXID	*Transaction Identifier*
	Identificador de Transacción
TXT	*Text*
	Texto
UTXO	*Unspent Transaction Output*
	Salida de Transacción sin Gastar
VA	*Validation Authority*
	Autoridad de Validación

CAPÍTULO 1

INTRODUCCIÓN

Science never solves a problem without creating ten more.

—GEORGE BERNARD SHAW

Los avances de las últimas décadas en la industria de la tecnología y computación han hecho posible el surgimiento del Internet, el comercio en línea y la banca electrónica; esto es, han generado alternativas para el intercambio monetario. El número de usuarios de servicios de pago en línea así como de banca móvil se ha ido incrementando en los últimos años. Pero el tener un ente central el cual emite y almacena las transacciones, como lo son los bancos o los servicios en línea como PayPal siempre ha generado controversia. La gente debe de confiar en estos terceros y dado que funcionan de manera centralizada, cualquier compromiso en su infraestructura puede resultar en la pérdida de información. Han existido múltiples reportes de robo de la información de los cuentahabientes así como de números de tarjetas de crédito de los bancos y de otros servicios, lo cual impacta a las personas en su privacidad y seguridad.

La centralización no sólo concentra la confianza sino la información y los procesos, lo cual puede derivar en grandes pérdidas en el caso de una violación de la seguridad.

El tener Internet como una gran plataforma donde todo mundo tiene acceso ha facilitado el poder distribuir información. Gracias a ello muchas tecnologías distribuidas y descentralizadas se pueden beneficiar de

esta capacidad [Beikverdi and Song, 2015].

En enero de 2009 Satoshi Nakamoto liberó la primera implementación de una criptomoneda descentralizada llamada Bitcoin[1]. El diseño de Bitcoin fue descrito por primera vez en un artículo autopublicado en Octubre de 2008 [Nakamoto, 2008].

Desde su invención, Bitcoin ha ganado gran popularidad así como atención de los medios. Para la comunidad criptográfica y de seguridad de la información, la idea de una criptomoneda no es nueva en lo absoluto. Desde 1982 David Chaum ya había realizado el diseño de un esquema anónimo de dinero electrónico, el cual presenta en [Chaum, 1983]. Desde entonces han sido publicados cientos de artículos académicos que mejoran la eficiencia y seguridad de los esquemas de dinero electrónico.

De forma natural surge una pregunta interesante: ¿Por qué a pesar de tres décadas de investigación en dinero electrónico, estos esquemas no se popularizaron, mientras que Bitcoin —un sistema diseñado e implementado posiblemente por una persona, la cual era totalmente desconocida, y que no emplea los avances criptográficos más recientes— ha logrado tener un gran éxito?

Aunque Bitcoin no emplea los avances criptográficos más recientes posee un diseño el cual refleja una sorprendente cantidad de ingenio y sofisticación. Pero aún más importante que esto es el que ataca los problemas principales de forma expedita [Barber et al., 2012].

1.1. Objetivos

El objetivo general del presente trabajo de tesis consiste en investigar el fenómeno de Bitcoin para obtener un conocimiento profundo de los mecanismos que se emplean para lograr asegurar sus transacciones en un medio público y con la presencia de adversarios maliciosos. Evaluando sus fortalezas y debilidades. Así como trasladar la aplicación de los conceptos implementados por Bitcoin para aprovecharlos en una propuesta de solución a otro problema con décadas de trabajo en el área de la seguridad de la información: la emisión y manejo de certificados de llave pública.

Como objetivos específicos planteados para esta tesis se encuentran:

- Dotar al lector de las herramientas teóricas para una completa apreciación del funcionamiento de Bitcoin.
- Proporcionar un material de referencia en español respecto al funcionamiento de Bitcoin.
- Dar una perspectiva más allá de las implicaciones técnicas que trae el fenomeno de Bitcoin.
- Realizar una propuesta de alternativa de infraestructura de llave pública.

[1]Dado que Bitcoin es a la vez una criptomoneda y un protocolo su escritura se puede prestar a confusiones. La escritura generalmente aceptada es usar Bitcoin (en singular y la primera letra con mayúscula) para referirse al protocolo, el software y la comunidad; mientras que bitcoin (en minúscula) se emplea para referirse a la unidad de la criptomoneda.

1.2. Estructura del documento

El presente documento se divide en ocho capítulos principales:

1. Contiene la presente introducción.
2. Se abordan los conceptos básicos de seguridad de la información que serán empleados a lo largo de los demás capítulos.
3. Se da una breve introducción a la criptografía, estos conceptos son necesarios para entender los capítulos 4, 6 y 7.
4. Se explica el uso de los certificados, su emisión, los modelos de confianza y el estándar X.509 que se emplea en infraestructuras de llave pública.
5. Se describe el funcionamiento del sistema de nombres de dominio (DNS), está información será necesaria para entender el capítulo 7.
6. Se presenta el concepto de dinero electrónico y sus implicaciones. Posteriormente, se describe a detalle el funcionamiento de Bitcoin y los mecanismos que emplea para asegurar sus transacciones. Finalmente, se presentan las implicaciones que ha traído la popularidad de Bitcoin en diferentes ámbitos.
7. A partir de los conceptos presentados en todos los capítulos previos se presenta una propuesta de alternativa de infraestructura de llave pública descentralizada.
8. Se exponen las conclusiones del trabajo realizado así como el trabajo futuro.

Adicional a lo contenido en los siete capítulos principales se incluyen dos anexos: el primero (A), con la referencia de las instrucciones del lenguaje *Script*, el cual es empleado por Bitcoin en la realización de sus transacciones. Y el segundo (B), con el esquema de base de datos y la interfaz de usuario del sistema de pago presentado en el capítulo 6.

CAPÍTULO 2

SEGURIDAD COMPUTACIONAL

Safety is something that happens between your ears, not something you hold in your hands.

—JEFF COOPER

El ambiente que predominaba en el pasado, en el que los sistemas operaban de manera aislada o en redes privadas, ha sido sustituido por computadoras personales, tabletas y teléfonos inteligentes, que cada vez tienen mayor capacidad de cómputo, así como por tecnologías convergentes y por la propia difusión masiva del uso de Internet. Hoy en día los participantes se encuentran cada vez más interconectados y esta interconexión se extiende más allá de las fronteras nacionales. Al mismo tiempo, Internet forma parte de la infraestructura operativa de sectores estratégicos como energía, transportes y finanzas, y desempeña un papel fundamental en la forma en que las compañías realizan sus transacciones comerciales, los gobiernos proporcionan sus servicios a los ciudadanos y a las empresas, y los ciudadanos se comunican e intercambian información de manera individual. La naturaleza y el tipo de tecnologías que constituyen la infraestructura de la información y comunicaciones también han cambiado de manera significativa. El número y el tipo de dispositivos que integran la infraestructura de acceso se ha multiplicado, incluyendo elementos de tecnología fija, inalámbrica y móvil, así como una proporción creciente de accesos que están conectados de manera permanente. Como consecuencia de todos estos cambios, la naturaleza, volumen y sensibilidad de la información que se intercambia a través de esta infraestructura se ha incrementado de manera muy significativa.

Como resultado de una creciente interconexión, los sistemas y las redes de información son más vulnerables, ya que están expuestos a un número creciente, así como a una mayor variedad, de amenazas y de vulnerabilidades. Esto hace que en materia de seguridad surjan nuevos retos que deben abordarse [OECD, 2002].

2.1. Definición de seguridad computacional

El Instituto Nacional de Estándares y Tecnología de los Estados Unidos de América (NIST, por sus siglas en inglés) define el término «seguridad computacional» de la siguiente manera [Guttman and Roback, 1995]:

> La protección proporcionada a un sistema automatizado de información con el fin de alcanzar los objetivos de preservar la integridad, disponibilidad y confidencialidad de los recursos del sistema de información (incluyendo hardware, software, firmware, información/datos y telecomunicaciones).

Dentro de esta definición se encuentran tres conceptos clave para la seguridad de la información: confidencialidad, integridad y disponibilidad. Normalmente conocidos como la *triada CIA* o *triada de la Seguridad*.

NIST define estos objetivos, así como la pérdida de seguridad en cada uno, de la siguiente manera [NIST, 2004]:

- ✓ **Confidencialidad**: preservar las restricciones autorizadas en el acceso y divulgación de la información, incluyendo medios para proteger la privacidad personal y la información propietaria. La pérdida de confidencialidad es la publicación no autorizada de información.
- ✓ **Integridad**: proteger la información de modificaciones inapropiadas o de su destrucción, incluyendo asegurar el no repudio y la autenticidad[1] de la información. La pérdida de integridad es la modificación o destrucción no autorizada de información.
- ✓ **Disponibilidad**: asegurar el acceso expedito y confiable a la información y su uso. La pérdida de disponibilidad es la disrupción al acceso o uso de la información, o de un sistema de información.

2.2. Arquitectura de seguridad de OSI

Para evaluar de manera efectiva los requerimientos de seguridad de una organización, así como para evaluar productos y políticas de seguridad, el administrador responsable de la seguridad requiere de una metodología para determinar los requerimientos de seguridad y para caracterizar las estrategias que permitan satisfacer tales requerimientos.

La recomendación X.800, «Arquitectura de Seguridad de OSI»[2] [CCITT, 1991], de la Unión Internacional de Telecomunicaciones (ITU, por sus siglas en inglés), provee una metodología adecuada, además de que proporciona una descripción de varios de los conceptos de seguridad computacional empleados en esta tesis. La arquitectura de seguridad de OSI se enfoca en ataques a la seguridad, así como en mecanismos y servicios de seguridad. Estos pueden ser definidos de la siguiente manera [Stallings, 2013]:

- **Ataques a la seguridad**: cualquier acción que comprometa la seguridad de la información perteneciente a una organización.

[1] Los términos *no repudio* y *autenticidad* mencionados forman parte de los «servicios de seguridad», mismos que serán descritos más adelante.

[2] La arquitectura de seguridad de OSI fué concebida en el contexto de la arquitectura del protocolo OSI, sin embargo, para la comprensión de este trabajo no es necesario tener un conocimiento previo del protocolo.

- **Mecanismos de seguridad**: un proceso (o un dispositivo con el proceso incorporado) diseñado para detectar, prevenir y recuperarse de una ataque a la seguridad.

- **Servicios de seguridad**: un servicio de procesamiento o de comunicación proporcionado por un sistema para dar un tipo especial de protección a los recursos del sistema; los servicios de seguridad implementan políticas de seguridad y son implementados a su vez, por mecanismos de seguridad.

2.2.1. Ataques a la seguridad

Una manera práctica de clasificar los ataques a la seguridad, empleada tanto en la recomendación X.800 [CCITT, 1991], como en el RFC 4949 [Shirey, 2007], es separarlos en ataques pasivos y activos.

Dramatis Personae

En la literatura de seguridad de la información y criptografía, normalmente, al describir los protocolos criptográficos, se emplean los nombres de los siguientes personajes:

1. **Alicia**: Primer participante en todos los protocolos.
2. **Beto**: Segundo participante en todos los protocolos.
3. **Eva**: Escucha los mensajes.
4. **Malena**: Atacante activo malicioso.

Ataques pasivos

En los ataques pasivos se escucha o supervisa el medio de comunicación (véase figura 2.1). El objetivo del oponente es obtener la información que se está transmitiendo. Son dos tipos de ataques pasivos: la obtención de información y el análisis de tráfico.

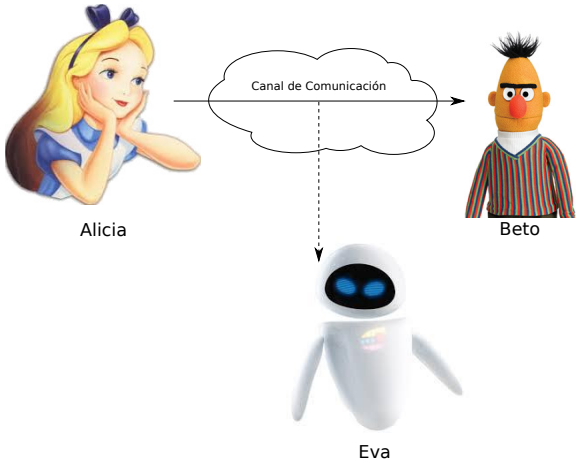


Figura 2.1: Ataque pasivo

La obtención de información es fácil de entender: de cualquier tipo de comunicación se pueden obtener datos sensibles o confidenciales. Queremos evitar que el oponente obtenga estos datos de la comunicación.

El análisis de tráfico es más sutil. Supongamos que protegemos la información de alguna manera y el oponente no puede entenderla —la manera usual de lograr esto es empleando algoritmos criptográficos—. Aunque la información no se pueda entender, el oponente puede observar los patrones, la frecuencia y el tamaño de los mensajes; también puede obtener la localización e identidad de los equipos que participan en la comunicación. Todo esto le puede ser de ayuda en determinar el tipo de mensajes que se están comunicando.

Los ataques pasivos son muy difíciles de detectar, debido a que no hay alteración a los datos. Típicamente los mensajes son enviados y recibidos de forma aparentemente normal, ni el destinatario ni el remitente se dan cuenta de que han sido observados.

Ataques activos

Los ataques activos involucran modificaciones a los datos o la creación de datos falsos, y pueden ser catalogados de la siguiente manera:

1. **Suplantación**: una entidad pretende ser alguien más. En la figura 2.2, Malena realiza la comunicación *b* diciendo ser Alicia.
2. **Retransmisión**: incluye la captura pasiva de datos y su subsecuente retransmisión para producir un efecto no autorizado. En la figura 2.2, ocurren las comunicaciones *c* y *a* primero, y posteriormente Malena retransmite lo mismo en la comunicación *b*.
3. **Modificación del mensaje**: alguna parte de un mensaje legitimo es alterada, retrasada o reordenada. En la figura 2.2, ocurre la comunicación *a*, Malena la modifica y posteriormente ocurre la comunicación *b*.
4. **Denegación de servicio**: inhibe el uso o administración de las comunicaciones. En la figura 2.2, la comunicación *c* es interrumpida.

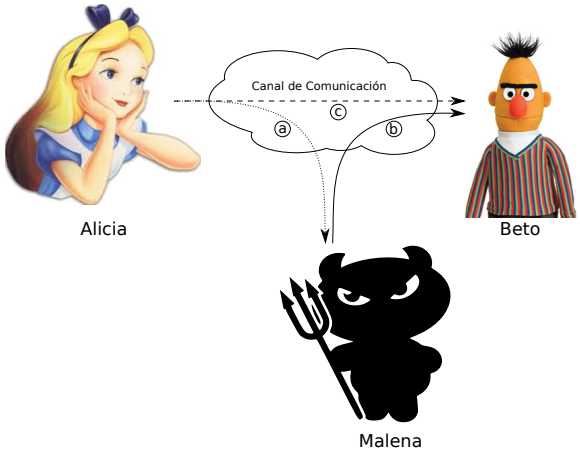


Figura 2.2: Ataques activos

2.2.2. Servicios de seguridad

La recomendación X.800 [CCITT, 1991] de la ITU define los servicios de seguridad como:

> Servicio proporcionado por una capa de sistemas abiertos comunicantes, que garantiza la seguridad adecuada de los sistemas y de la transferencia de datos.

Quizá una definición más entendible es la proporcionada en el RFC 4949 [Shirey, 2007]:

> Un servicio de procesamiento o comunicación que es provisto por un sistema para proporcionar un tipo específico de protección a los recursos del sistema.

Así mismo, en la citada recomendación, los servicios de seguridad se dividen en cinco categorías más:

1. **Autenticación**: a este servicio le concierne asegurar que una comunicación es auténtica. En el caso de un mensaje, la función del servicio de autenticación es asegurar al receptor que el mensaje proviene realmente del remitente que se indica. En el caso de una comunicación, su función es verificar que la persona con la que se estableció la comunicación sea quien dice ser.

2. **Control de acceso**: es la habilidad de limitar y controlar el acceso a un sistema y sus aplicaciones.

3. **Confidencialidad de los datos**: se refiere a la propiedad de que los datos o información no se encuentren disponibles, o sean revelados a entidades o procesos no autorizados.

4. **Integridad de los datos**: verifica que los datos no han sido alterados por una entidad no autorizada, lo cual significa, que el destinatario de la información sea capaz de detectar si hubo anomalías en la transmisión.

5. **No repudio**: previene que cualquiera de las partes niegue haber transmitido un mensaje.

2.2.3. Mecanismos de seguridad

La recomendación X.800 [CCITT, 1991] divide en dos categorías los mecanismos de seguridad:

1. Mecanismos de seguridad específicos.

2. Mecanismos de seguridad generales.

Mecanismos de seguridad específicos

Son mecanismos que pueden incorporarse en la capa (N) apropiada para proporcionar algunos de los servicios de seguridad de OSI:

- **Cifrado**: el cifrado puede proporcionar la confidencialidad de la información de datos o del flujo de tráfico, y puede desempeñar una función en varios otros mecanismos de seguridad o complementarlos. Los algoritmos de cifrado pueden ser reversibles o irreversibles.

- **Mecanismo de firma digital**: estos mecanismos definen dos procedimientos:

  1. Firma de una unidad de datos.
  2. Verificación de una unidad de datos firmada.

El primer proceso emplea información del firmante que es privada (es decir, única y confidencial). El segundo proceso se vale de algoritmos que están disponibles públicamente, pero a partir de los cuales no se puede deducir cuál es la información privada del firmante.

- **Mecanismo de control de acceso**: estos mecanismos pueden utilizar la identidad autenticada, las capacidades o la información acerca de una entidad, para determinar y aplicar los derechos de acceso correspondientes a la misma. Si la entidad intenta consumir un recurso no autorizado, o un recurso autorizado con un tipo impropio de acceso, la función de control de acceso rechazará la tentativa y puede además informar del incidente con los efectos de generar una alarma y/o anotarlo en una bitácora de auditoría de seguridad.
- **Mecanismo de integridad de los datos**: la integridad de los datos tiene dos aspectos: la integridad de una sola unidad de datos o de un solo campo, y la integridad de un tren de unidades de datos o de campos de unidad de datos. En general, se utilizan diferentes mecanismos para proporcionar estos dos tipos de servicios de integridad, aunque no es práctica la provisión del segundo sin el primero.
- **Mecanismo de intercambio de autenticación**: algunas de las técnicas que pueden aplicarse a los intercambios de autenticación son:
  1. Utilización de información de autenticación, como contraseñas, suministradas por una entidad expedidora y verificadas, por la entidad receptora.
  2. Técnicas criptográficas.
  3. Uso de características y/o propiedades de la entidad.
- **Mecanismo de relleno de tráfico**: pueden utilizarse mecanismos de relleno de tráfico para proporcionar diversos niveles de protección contra análisis del tráfico. Este mecanismo puede ser eficaz únicamente si el relleno de tráfico está protegido por un servicio de confidencialidad.
- **Mecanismo de control de encaminamiento**: las rutas pueden elegirse dinámicamente o por acuerdo previo con el fin de utilizar sólo subredes, conmutadores o enlaces físicamente seguros.
- **Mecanismo de notarización**: pueden garantizarse las propiedades sobre los datos comunicados entre dos o más entidades, tales como su integridad, origen, fecha y destino, mediante la provisión de un mecanismo de notarización. La seguridad es proporcionada por una tercera parte que actúa como notario, en la cuál las entidades comunicantes confían y que mantiene la información necesaria para proporcionar la garantía requerida de una manera verificable.

Mecanismos de seguridad generales

Consisten de mecanismos que no son específicos a un servicio particular. Por tanto, no forman parte de una capa determinada. En general, la importancia de estos mecanismos está directamente relacionada con el nivel de seguridad requerido.

- **Funcionalidad de confianza**: puede utilizarse para ampliar el campo de aplicación o para establecer la eficacia de otros mecanismos de seguridad. Es determinada mediante otros mecanismos, como políticas de seguridad.
- **Etiquetas de seguridad**: los recursos que comprenden elementos de datos pueden tener asociadas etiquetas de seguridad, por ejemplo, para indicar un nivel de sensibilidad.

- **Detección de eventos**: abarca la detección de violaciones aparentes de seguridad y puede incluir también la detección de eventos normales.
- **Registro de auditoria de seguridad**: proporciona un mecanismo de seguridad valioso dado que hacen posible detectar e investigar las violaciones de seguridad potenciales, permitiendo una auditoría de seguridad posterior.
- **Recuperación de seguridad**: trata las peticiones provenientes de mecanismos tales como las funciones de tratamiento y de gestión de los eventos, y realiza acciones de recuperación como resultado de la aplicación de un conjunto de reglas.

2.2.4. Relación entre servicios y mecanismos de seguridad

En la tabla 2.1 se muestran los mecanismos, solos o combinados con otros, que a veces se consideran apropiados para suministrar cada servicio.

	Cifrado	Firma digital	Control de acceso	Integridad de datos	Intercambio de autenticación	Relleno de tráfico	Control de encaminamiento	Notarización
Autenticación	✓	✓	-	-	✓	-	-	-
Control de acceso	-	-	✓	-	-	-	-	-
Confidencialidad	✓	-	-	-	-	✓	✓	-
Integridad	✓	✓	-	✓	-	-	-	-
No repudio	-	✓	-	✓	-	-	-	✓

Tabla 2.1: Mecanismos de seguridad apropiados para cada servicio de seguridad

CAPÍTULO 3

CRIPTOGRAFÍA

I am fairly familiar with all forms of secret writings, and am myself the author of a trifling monograph upon the subject, in which I analyze one hundred and sixty separate ciphers, but I confess that this is entirely new to me. - Sherlock Holmes

—SIR ARTHUR CONAN DOYLE, The Adventure of the Dancing Men

La criptografía es una rama de la criptología. Esta se encarga del diseño y el análisis de técnicas matemáticas que permitan establecer comunicaciones seguras ante la presencia de adversarios maliciosos. La criptografía se divide en tres ramas principales [Paar and Pelzl, 2010]:

1. **Algoritmos simétricos**: también denominados de llave privada, en los cuales dos entidades tienen un método de cifrado y descifrado para el cual comparten una llave secreta. Toda la criptografía existente antes de 1976 estaba basada exclusivamente en algoritmos simétricos.

2. **Algoritmos asimétricos**: también llamados de llave pública, fueron propuestos en 1976 por Whitfield Diffie, Martin Hellman [Diffie and Hellman, 1976] y Ralph Merkle [Merkle, 1978]. En la criptografía de llave pública el usuario posee una llave secreta, pero también posee una llave pública. Los algoritmos asimétricos pueden ser usados para aplicaciones como firma digital, acuerdo de llave y cifrado.

3. **Protocolos criptográficos**: explicado *grosso modo*, los protocolos criptográficos lidian con la implementación de comunicaciones seguras, tomando a los algoritmos simétricos y asimétricos como bloques básicos.

3.1. Criptografía simétrica

Para explicar de forma intuitiva la criptografía simétrica se puede hacer una analogía como se muestra en la figura 3.1. Imaginemos que se tiene una caja fuerte con una cerradura muy buena. La llave sólo la poseen dos personas, Alicia y Beto. En esta analogía, la acción de que Alicia cifre un mensaje es el equivalente a que ella coloque el mensaje en la caja fuerte y la cierre. Para poder descifrar el mensaje Beto usa su llave y abre la caja fuerte.

Figura 3.1: Analogía de un cifrado simétrico: una caja fuerte con una cerradura

3.1.1. Distribución de llaves

En criptografía simétrica la llave debe de ser acordada entre Alicia y Beto usando un canal seguro. Siempre se supone que el medio de comunicación de los mensajes está siendo escuchado por adversarios maliciosos, por lo tanto la llave no puede ser enviada a través de éste.

Aun teniendo un medio seguro para acordar las llaves se presenta otro problema: debido a que todas las comunicaciones se hacen uno a uno, se requiere lidiar con una cantidad muy grande de llaves, ya que cada par de usuarios requiere de un par de llaves diferente. Teniendo una red con n usuarios, se requieren

$$\binom{n}{2} = \frac{n(n-1)}{2}$$

pares de llaves y cada quien debe almacenar de manera segura $n-1$ llaves.

3.1.2. No repudio

Al tener Alicia y Beto las mismas capacidades, ya que ambos poseen la misma llave, se tiene por consecuencia de que la criptografía simétrica no pueda ser empleada en aplicaciones donde se desee evitar que Alicia o Beto hagan trampa. Por ejemplo, Alicia podría mandar un mensaje incriminatorio y al enfrentarse a una posible sanción, ella podría argumentar que Beto maliciosamente lo creó.

3.1.3. Funciones picadillo

Las funciones picadillo, también llamadas funciones *hash* o funciones de digestión, a diferencia de los algoritmos descritos previamente, no emplean una llave. En ellas, se calcula el *digesto* de un mensaje, que es una cadena de bits corta y de tamaño fijo. Para un mensaje en particular, el digesto del mensaje puede ser visto como su huella digital. Existen tres propiedades principales que las funciones picadillo deben de poseer para ser seguras [Paar and Pelzl, 2010]:

1. **Resistente a preimagen**: deben tener sólo un sentido, es decir dada la salida z debe de ser computacionalmente inviable el poder encontrar un mensaje x tal que $z = h(x)$.
2. **Resistente a segunda preimagen**: se refiere a que debe de ser computacionalemente inviable el poder crear dos mensajes diferentes $x_1 \neq x_2$ tales que produzcan el mismo picadillo $z_1 = h(x_1) = h(x_2) = z_2$. En esta propiedad se presupone que x_1 ya está dado y es fijo.
3. **Resistente a colisiones**: se refiere a que debe de ser computacionalemente inviable el poder crear dos mensajes diferentes $x_1 \neq x_2$ tales que produzcan el mismo picadillo $h(x_1) = h(x_2)$. En esta propiedad se presupone que x_1 y x_2 pueden ser elegidos arbitrariamente.

Ser o no ser

Dependiendo del autor, las funciones picadillo pueden ser consideradas como una rama de la criptografía o encontrarse contenidas en la criptografía simétrica.

3.2. Criptografía asimétrica

El concepto de criptografía asimétrica surgió del esfuerzo de solucionar dos de los principales problemas asociados a la criptografía simétrica y que fueron explicados previamente: la distribución de llaves y el no repudio.

Los algoritmos asimétricos emplean una llave para cifrar y otra llave totalmente diferente pero relacionada para descifrar. Estos algoritmos poseen la siguiente característica [Stallings, 2013]:

> Es computacionalmente inviable determinar la llave de descifrado teniendo acceso al algoritmo criptográfico y a la llave de cifrado.

3.2.1. Cifrado y descifrado

Retomando la analogía de la caja fuerte, en el escenario de la criptografía asimétrica existen los siguientes cambios: Alicia posee su llave privada y pública, su llave pública permite colocar mensajes en la caja fuerte pero no sacarlos; la llave privada permite sacar los mensajes dentro de la caja fuerte, ver figura 3.2.

3.2.2. Distribución de llaves

En el caso de la criptografía asimétrica no es necesario establecer pares de llaves entre cada par de individuos. Si algún usuario A desea enviar un mensaje a otro usuario B, basta con que B consiga la llave pública de A. Por lo que en comparación con la criptografía simétrica, para una red de n usuarios, solo son necesarios n pares de llaves (pública y privada de cada usuario).

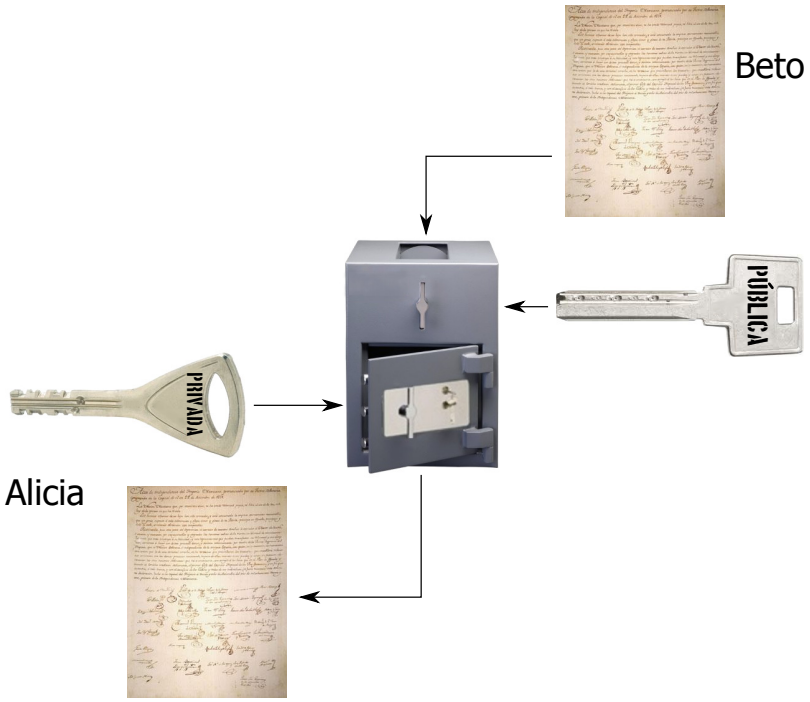


Figura 3.2: Analogía de un cifrado asimétrico: una caja fuerte con buzón rotativo

Continuando con la analogía, Alicia puede dejar colgada en la pared junto a la caja fuerte su llave pública, así cualquiera puede depositarle (cifrar) mensajes, o, en su defecto proporcionarle una copia a quien se la solicite. Cuando ella quiere leer los mensajes (descifrar) emplea su llave privada para abrir la caja fuerte y los extrae.

3.2.3. No repudio

Con criptografía asimétrica es posible probar que un usuario emitió algún mensaje. Esto se logra empleando algoritmos conocidos como de firma digital.

Firma digital

La firma autógrafa ha sido empleada desde hace mucho como prueba de la autoría o consentimiento del contenido de un documento. Lo que más nos importa de una firma es:

1. **Que la firma sea auténtica**: la firma convence al receptor del documento que el firmante deliberadamente lo firmó.
2. **Que la firma sea infalsificable**: la firma es la prueba de que el firmante, y no alguien mas, deliberadamente lo firmó.
3. **Que la firma no sea reusable**: la firma es parte del documento, no puede ser transportada a un documento diferente.

4. **Que un documento firmado es inalterable**: después de que un documento es firmado, este no puede ser alterado.

5. **Que la firma no pueda ser repudiada**: la firma y el documento son objetos físicos. El signatario no puede posteriormente alegar que él no firmó.

En el mundo real ninguna de estas aseveraciones es totalmente cierta, pero estamos dispuestos a vivir con esto debido a la dificultad de hacer trampa y el riesgo de ser descubierto [Schneier, 1995]. Sin embargo al trasladar la firma al mundo digital, las cinco aseveraciones pueden satisfacerse por completo empleando criptografía asimétrica.

La firma digital es una primitiva criptográfica fundamental para los servicios de autenticación, integridad de datos y no repudio. El propósito de una firma digital es proveer un medio para asociar a una entidad con una pieza de información. La firma digital de un mensaje es un número que depende de un secreto conocido únicamente por el firmante (llave privada), así como, del contenido del mensaje firmado. Las firmas deben ser verificables por cualquiera, sin necesidad de conocer la llave privada del firmante [Menezes et al., 1996]. El esquema básico de firma digital se muestra a continuación en la figura 3.3:

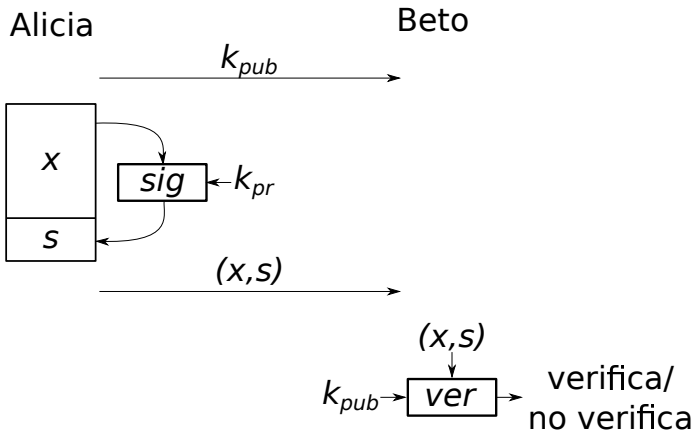


Figura 3.3: Principio de una firma digital

En la figura 3.3 se puede apreciar el proceso de firma y verificación. En un primer momento Alicia envía su llave pública (k_{pub}) a Beto, con la cual él verificará la firma. Posteriormente cuando Alicia quiere firmar un mensaje, ella emplea el algoritmo de firma sig, el cual requiere como entrada el mensaje a firmar (x) y la llave privada del firmante (k_{pr}). La salida del algoritmo sig es la firma digital del mensaje (s). Alicia envía el mensaje original junto como la firma generada (x, s) a Beto. Finalmente Beto emplea el algoritmo de verificación ver, este requiere como entradas la llave pública del firmante (k_{pub}), el mensaje (x) y la firma del mensaje (s). El algoritmo ver entregará como resultado si la firma *verifica* o no.

3.2.4. Autenticidad de las llaves

A partir de lo que se ha descrito previamente, se puede ver que la mayor ventaja de la criptografía asimétrica es que se pueden distribuir llaves públicas de forma indiscriminada. Pero, en la práctica esto tiene sus bemoles, porque debemos de asegurar la autenticidad de las llaves públicas.

Ataque del intruso de en medio

Para entender el porqué es importante la autenticidad de las llaves públicas, continuaremos con la analogía de la caja fuerte, sólo que ahora tenemos muchas más, digamos miles de cajas, todas se encuentran en un recinto donde al entrar a la recepción se encuentra un tablero con las llaves públicas de todas las cajas; las llaves tienen un número y existe una lista impresa que indica a quién le pertenece cada llave.

Incorporaremos ahora a otro personaje: Eva. Eva es una persona que quiere enterarse de los mensajes que se intercambian Alicia y Beto. Para lograr esto, Eva tiene dos cajas fuertes y sabe que la lista de llaves públicas no es muy vigilada, por lo que puede sustituirla fácilmente. Para lograr su objetivo, ella sustituye la lista original con una en la que cambió los números de llave pública de Alicia y de Beto con los números de las llaves públicas de cada una de las cajas que ella posee.

Cuando Beto quiere dejarle un mensaje a Alicia revisa la lista, toma la llave y deposita el mensaje en esa caja fuerte. A continuación, Eva, que posee la llave privada, saca el mensaje, lo lee y, posteriormente, lo coloca en la verdadera caja fuerte de Alicia. Lo mismo pasa cuando Alicia quiere comunicarse con Beto. Ni Alicia ni Beto se enteran de que alguien más está leyendo sus mensajes. Es más, Eva podría tener intenciones más perversas, como alterar los mensajes. A este escenario se le conoce como el ataque del intruso de en medio (MitM, por sus siglas en inglés).

3.3. Protocolos criptográficos

Un protocolo criptográfico es un algoritmo distribuido, conformado de una secuencia de pasos que especifica, de manera precisa, las acciones requeridas para dos o más entidades con el fin de lograr un objetivo especifico de seguridad. Los protocolos juegan un papel importante en la criptografía y son esenciales para lograr ofrecer los servicios de autenticidad, confidencialidad, integridad y no repudio [Menezes et al., 1996]. Para construir un protocolo criptográfico se pueden usar esquemas de cifrado, firmas digitales, funciones picadillo y generadores de números aleatorios entre otras primitivas.

Se dice que un protocolo es vulnerado cuando falla en lograr los objetivos para los cuales fue diseñado, de una manera tal en la que un adversario logra obtener ventaja no por romper alguna de las primitivas subyacentes de forma directa, como lo podría ser un esquema de cifrado, sino al manipular el protocolo. Un protocolo logra ser vulnerado por varias razones, entre las que están:

1. Debilidad en alguna primitiva criptográfica la cual puede ser amplificada por el protocolo.
2. Garantías de seguridad asumidas o prometidas las cuales fueron ya sea sobrestimadas o no fueron entendidas en su completitud.
3. El no considerar algunos principios aplicables a toda una clase de primitivas, como lo es el cifrado.

3.4. Criptografía de curvas elípticas

La criptografía de curvas elípticas, es en esencia, el uso del grupo de puntos en una curva elíptica como el sistema de numeración para criptografía de llave pública. Existen dos razones principales para emplear curvas elípticas como la base de criptosistemas de llave pública. La primera, se considera que los criptosistemas basados en curva elíptica brindan mejor seguridad que los criptosistemas tradicionales para un tamaño de llave dado. Se puede tomar ventaja de este hecho para incrementar la seguridad, o (más comúnmente) incrementar el desempeño al reducir el tamaño de la llave mientras se conserva el mismo nivel de seguridad.

La segunda razón es que la estructura de una curva elíptica puede ser empleada para construir criptosistemas con propiedades interesantes las cuales son difíciles o hasta imposibles de lograr de otra forma. Un ejemplo notable es la criptografía basada en identidad, así como el subsecuente surgimiento de los protocolos criptográficos basados en emparejamientos [Stavroulakis and Stamp, 2010].

3.4.1. Curvas elípticas

Explicado de forma burda, una curva elíptica es un conjunto de puntos sobre un campo que satisface una ecuación en dos variables: x y y, en la que la variable y es de grado 2 y la variable x de grado 3, a la cual se le puede asociar una ley de grupo. Además de los puntos en la forma (x, y), existe un punto extra, denotado como ∞, el cuál sirve como el elemento identidad del grupo.

Definición

Una curva elíptica E sobre un campo K está definida por la ecuación de Weierstrass [Hankerson et al., 2003]:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{3.1}$$

donde $a_1, a_2, a_3, a_4, a_6 \in K$ y $\Delta \neq 0$, conocido como el *discriminante* de E, es definido de la siguiente manera:

$$\left.\begin{aligned} \Delta &= -d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6 \\ d_2 &= a_1^2 + 4a_2 \\ d_4 &= 2a_4 + a_1a_3 \\ d_6 &= a_3^2 + 4a_6 \\ d_8 &= a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2 \end{aligned}\right\} \tag{3.2}$$

Si L es cualquier extensión del campo K, entonces el conjunto de los *puntos L-racionales* en E[1] es:

$$E(L) = \{(x, y) \in L \times L : y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6 = 0\} \cup \{\infty\}$$

donde ∞ es el *punto al infinito*.

Ecuaciones de Weierstrass simplificadas

Dos curvas elípticas E_1 y E_2 definidas sobre K y dadas por las ecuaciones de Weierstrass

$$E_1 : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$
$$E_2 : y^2 + \bar{a}_1xy + \bar{a}_3y = x^3 + \bar{a}_2x^2 + \bar{a}_4x + \bar{a}_6$$

se dice que son *isomorfas* sobre K si existe $u, r, s, t \in K, u \neq 0$, de tal forma que el cambio de variables

$$(x, y) \rightarrow (u^2x + r, u^3y + u^2sx + t) \tag{3.3}$$

transforma la ecuación E_1 en la ecuación E_2. La transformación 3.3 se conoce como un *cambio admisible de variables*.

[1] Los puntos que satisfacen E sobre el campo L

Una ecuación de Weierstrass

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

definida sobre K puede ser simplificada de forma considerable al aplicar cambios admisibles de variables. Se consideran de forma separada los casos donde el campo subyacente K tiene característica diferente de 2 y 3, o tiene característica 2 o 3.

1. Si la característica de K es diferente de 2 y 3, entonces el cambio admisible de variables

$$(x, y) \to \left(\frac{x - 3a_1^2 - 12a_2}{36}, \frac{y - 3a_1x}{216} - \frac{a_1^3 + 4a_1a_2 - 12a_3}{24}\right)$$

transforma E en la curva

$$y^2 = x^3 + ax + b \tag{3.4}$$

donde $a, b \in K$. El discriminante de esta curva es $\Delta = 16(4a^3 + 27b^2)$.

2. Si la característica de K es 2, entonces existen dos casos a considerar. Si $a_1 \neq 0$, entonces el cambio admisible de variables

$$(x, y) \to \left(a_1^2x + \frac{a_3}{a_1}, a_1^3y + \frac{a_1^2a_4 + a_3^2}{a_1^3}\right)$$

transforma E en la curva

$$y^2 + xy = x^3 + ax^2 + b \tag{3.5}$$

donde $a, b \in K$. Este tipo de curva se conoce como *no supersingular* y tiene un discriminante $\Delta = b$. Si $a_1 = 0$, entonces el cambio admisible de variables

$$(x, y) \to (x + a_2, y)$$

transforma E en la curva

$$y^2 + cy = x3 + ax + b \tag{3.6}$$

donde $a, b, c \in K$. Este tipo de curva se conoce como *supersingular* y tiene un discriminante $\Delta = c^4$.

3. Si la característica de K es 3, entonces existen dos casos a consideras. Si $a_1^2 \neq -a_2$, entonces el cambio admisible de variables

$$(x, y) \to \left(x + \frac{d_4}{d_2}, y + a_1x + a_1\frac{d4}{d2} + a_3\right),$$

donde $d_2 = a_1^2 + a_2$ y $d_4 = a_4 - a_1a_3$, transforma E en la curva

$$y^2 = x^3 + ax^2 + b \tag{3.7}$$

donde $a, b \in K$. Este tipo de curva se conoce como *no supersingular* y tiene un discriminante $\Delta = -a^3b$. Si $a_1^2 = -a_2$, entonces el cambio admisible de variables

$$(x, y) \to (x, y + a_1x + a_3)$$

transforma E en la curva

$$y^2 = x^3 + ax + b \tag{3.8}$$

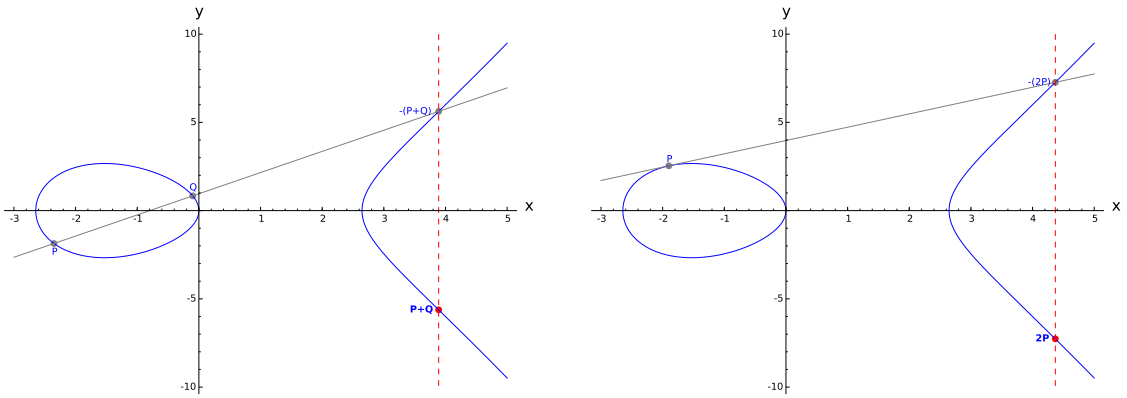
Este tipo de curva se conoce como *supersingular* y tiene un discriminante $\Delta = -a^3$.

Ley de grupo

Sea E una curva elíptica definida sobre el campo K. Hay una *regla de cuerda y tangente* para sumar dos puntos en $E(K)$ para obtener un tercer punto en $E(K)$. Junto con esta operación de suma, el conjunto de puntos de $E(K)$ forma un grupo abelinao con ∞ fungiendo como su identidad. Este es el grupo que se emplea en la construcción de sistemas criptográficos de curvas elípticas.

La regla de suma se explica de mejor de manera en forma geométrica. Sen $P = (x_1, y_1)$ y $Q = (x_2, y_2)$ dos puntos distintos en una curva elíptica E. Entonces la *suma* R de P y Q está definida de la siguiente manera: primero, se dibuja una línea a través de P y Q; esta línea interseca a la curva elíptica en un tercer punto. Entonces, R es la reflexión de este punto con respecto al eje x. Esto se muestra en la figura 3.4a.

El *doblado* R, de P, está definido de la siguiente manera: primero, se dibuja una línea tangente a la curva elíptica en P. Esta línea interseca la curva elíptica en un segundo punto. Entonces R es la reflexión de este punto con respecto al eje x. Esto se muestra en la figura 3.4b.



(a) Suma (b) Doblado

Figura 3.4: Las operaciones de grupo en la curva elíptica $y^2 = x^3 - 7x$

Las formulas algebráicas para la ley de grupo se pueden derivar de la descripción geométrica. Estas formulas se presentan a continuación para las curvas elípticas E de las versiones simplificadas de la ecuación de Weierstrass de la forma 3.4 en coordenadas afines cuando la característica del campo K subyacente es diferente de dos y tres, y para curvas elípticas E no supersingulares de la forma 3.5 sobre $K = \mathbb{F}_{2^m}$.

Ley de grupo para $E/K : y^2 = x^3 + ax + b, char(K) \neq 2, 3$

1. Identidad. $P + \infty = \infty + P = P$ para toda $P \in E(K)$.

2. Negativo. Si $P = (x, y) \in E(K)$, entonces $(x, y) + (x, -y) = \infty$. El punto $(x, -y)$ se denota como $-P$ y es llamado el *negativo* de P; es importante mencionar que $-P$ es un punto en $E(K)$. También $-\infty = \infty$.

3. Suma. Sea $P = (x_1, y_1) \in E(K)$ y $Q = (x_2, y_2) \in E(K)$, donde $P \neq \pm Q$. Entonces $P + Q = (x_3, y_3)$, donde

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \quad \text{y} \quad y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1.$$

4. Doblado. Sea $P = (x_1, y_1) \in E(K)$, donde $P \neq -P$. Entonces $2P = (x_3, y_3)$, donde

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1 \quad \text{y} \quad y_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1.$$

Ley de grupo para $E/\mathbb{F}_{2^m} : y^2 + xy = x^3 + ax^2 + b$ no supersingulares

1. Identidad. $P + \infty = \infty + P$ para todo $P \in E(\mathbb{F}_{2^m})$.
2. Negativo. Si $P = (x, y) \in E(\mathbb{F}_{2^m})$, entonces $(x, y) + (x, x + y) = \infty$. El punto $(x, x + y)$ se denota como $-P$ y es llamado el *negativo* de P; es importante mencionar que $-P$ es un punto en $E(\mathbb{F}_{2^m})$. También $-\infty = \infty$.
3. Suma. Sea $P = (x_1, y_1) \in E(\mathbb{F}_{2^m})$ y $Q = (x_2, y_2) \in E(\mathbb{F}_{2^m})$, donde $P \neq \pm Q$. Entonces $P + Q = (x_3, y_3)$, donde

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a \quad \text{y} \quad y_3 = \lambda(x_1 + x_3) + x_3 + y_1$$

   con $\lambda = (y_1 + y_2)/(x_1 + x_2)$.
4. Doblado. Sea $P = (x_1, y_1) \in E(\mathbb{F}_{2^m})$, donde $P \neq -P$. Entonces $2P = (x_3, y_3)$, donde

$$x_3 = \lambda^2 + \lambda + a = x_1^2 + \frac{b}{x_1^2} \quad \text{y} \quad y_3 = x_1^2 + \lambda x_3 + x_3$$

   con $\lambda = x_1 + y_1/x_1$.

3.4.2. Problema del logaritmo discreto

Supongamos que $(G, \cdot)$ es un grupo cíclico de orden n con un generador g, y la operación binaria $\cdot$ emplea una representación multiplicativa. Si elegimos un número entero x de forma aleatoria en el intervalo $[1, n - 1]$ y realizamos la operación

$$y = g^x$$

el problema de determinar x dados g, n y y es conocido como el *problema del logaritmo discreto* en G.

Para que un sistema cuya seguridad descansa en el problema del logaritmo discreto en G sea eficiente es necesario conocer algoritmos rápidos que calculen la operación de grupo. Por seguridad el problema del logaritmo discreto en G debe de ser intratable en términos computacionales.

Cualesquiera dos grupos cíclicos del mismo orden n son esencialmente iguales; esto es, tienen la misma estructura aunque los elementos se escriban de forma diferente. Las diferentes representaciones de los

elementos del grupo pueden derivar en algoritmos de diferentes velocidades para calcular la operación de grupo, así como para resolver el problema del logaritmo discreto.

Los grupos más utilizados para implementar sistemas basados en el problema del logaritmo discreto son los subgrupos cíclicos con representación multiplicativa, asociados a un campo finito y los subgrupos cíclicos de una curva elíptica dada [Hankerson et al., 2003].

Problema del logaritmo discreto en curvas elípticas

Sea E una curva elíptica definida sobre un campo finito $\mathbb{F}_p$. Sea P un punto en $E(\mathbb{F}_p)$ y suponiendo que P tiene orden primo n. Entonces el subgrupo cíclico de $E(\mathbb{F}_p)$ generado por P es

$$\langle P \rangle = \{\infty, P, 2P, 3P, \ldots, (n-1)P\}.$$

Si se elige un número entero d de forma aleatoria en el intervalo $[1, n-1]$ y realizamos la operación

$$Q = dP$$

el problema de determinar d dados p, E, P, n y Q es conocido como el *problema del logaritmo discreto en curvas elípticas*.

Entre lo público y lo privado

En los diferentes esquemas y protocolos basados en el problema del logaritmo discreto los parámetros de dominio son g y n; y es la llave pública y x la llave privada. Para el caso de curva elíptica los parámetros de dominio son p, E, P y n; Q es la llave pública y d la llave privada.

3.4.3. Estándares de criptografía de curvas elípticas

La estandarización de protocolos criptográficos y en especial de protocolos que emplean curvas elípticas, ha avanzado de forma importante sobre todo en los últimos años. El proceso de estandarización es importante si se desea emplear a gran escala, ya que diferentes usuarios tendrán diferentes combinaciones de software/hardware. El trabajar en un estándar bien definido para cualquier tecnología ayuda en la interoperatividad, así como facilita la adopción de esta tecnología. En el contexto de criptografía de curvas elípticas, los estándares definen no sólo el funcionamiento exacto de cada algoritmo, sino también el formato de los datos transmitidos. Los siguientes estándares son de particular relevancia para la criptografía de curvas elípticas [Blake et al., 2005]:

- **IEEE 1363**: [IEEE, 2009] Contiene prácticamente todos los algoritmos de llave pública. En particular, incluye ECDH, ECDSA, ECMQV y ECIES. Además contiene un apéndice que cubre los algoritmos básicos de teoría de números que se requieren para criptografía de llave pública.
- **ANSI X9.62 y X9.63**: [ANSI, 2005] [ANSI, 2011] Estos dos estándares se enfocan en criptografía de curva elíptica, tratándose ECDSA en X9.62 y ECDH, ECMQV y ECIES en X9.63. En ellos se especifica el formato del mensaje, así como se da una lista de curvas recomendadas.
- **FIPS 186.4**: [NIST, 2013] Es un estándar del NIST, el cual ha sido revisado en cuatro ocasiones. Especifica los algoritmos de DSA y ECDSA, también proporciona un lista de curvas recomendadas. Es un estándar obligatorio para todas las oficinas de gobierno de los Estados Unidos de América.

- **SECG**: [SECG, 2009] Fue redactado por un grupo industrial liderado por la empresa Certicom. El contenido es esencialmente el de los estándares ANSI, pero a diferencia de estos, es gratuito y se puede descargar de la pagina `www.secg.org`.

- **ISO**: [ISO, 2002] [ISO, 2006] Existen dos estándares ISO relevantes: ISO 15946-2, el cual cubre ECDSA e ISO 18033-2, que incluye una variante de ECIES llamada ECIES-KEM.

ECDSA

Es la variante de curva elíptica del algoritmo de firma digital (DSA, por sus siglas en inglés), también conocido como el estándar de firma digital (DSS, por su siglas en inglés). Antes de presentar ECDSA se describirá DSA para mostrar que no es mas que una generalización.

En DSA primero se elige una función picadillo H que da como salida una cadena de m bits de longitud. Después se define un número primo p de n bits tal que

- q divide a p -1.

- El problema del logaritmo discreto en un subgrupo de $\mathbb{F}_p$ de orden q es intratable en términos computacionales.

Con las técnicas y tecnologías de cómputo actuales, el segundo inciso significa que n debe de ser por lo menos 1024. Así como para evitar ataques por la paradoja del cumpleaños en la función picadillo se debe de elegir un valor de m mayor a 160. Después se requiere encontrar un generador g para el subgrupo de orden q en $\mathbb{F}_p^*$. Esto se logra tomando elementos aleatorios $h \in \mathbb{F}_p^*$ y calculando

$$g = h^{(p-1)/q} \pmod{p}$$

hasta obtener un valor $g \neq 1$. La probabilidad de elegir un elemento que no sea generador en el primer intento es de $1/q$; por lo que es muy fácil hallar un generador g. Típicamente DSA emplea SHA-1 como función picadillo. La tupla (H, p, q, g) es conocida como los parámetros de dominio.
DSA emplea la función

$$f : \begin{cases} \mathbb{F}_p^* & \to \mathbb{F}_q \\ x & \mapsto x \pmod{q} \end{cases}$$

donde al realizar la reducción módulo q se interpreta a $x \in \mathbb{F}_p^*$ como un entero. Esta función se emplea para proyectar elementos del grupo a los enteros módulo q y es conocida normalmente como la función de conversión. Al ser DSA un algoritmo de criptografía asimétrica es necesario generar el par de llaves pública y privada (y, x) donde

$$y = g^x \pmod{p}.$$

Los algoritmos de firma y verificación con DSA son los siguientes:

Algoritmo Firma DSA

Entrada: Un mensaje m y la llave privada x.
Salida: Una firma (r, s) del mensaje m.

1: Elegir $k \in \mathbb{R}\{1, \ldots, q-1\}$
2: $t \leftarrow g^k \pmod{p}$
3: $r = f(t)$
4: **if** $r = 0$ **then** `goto` 1
5: **end if**
6: $e \leftarrow H(m)$
7: $s \leftarrow (e + xr)/k \pmod{q}$
8: **if** $s = 0$ **then** `goto` 1
9: **end if**
10: **return** (r, s)

Algoritmo Verificación de Firma DSA

Entrada: Un mensaje m , una firma (r, s) y la llave pública y.
Salida: Verifica / No Verifica.

1: **if** $r, s \notin \{1, \ldots, q-1\}$ **then**
2: **return** No Verifica
3: **end if**
4: $e \leftarrow H(m)$
5: $u_1 \leftarrow e/s \pmod{q}, u_2 \leftarrow r/s \pmod{q}$
6: $t \leftarrow g^{u_1} y^{u_2} \pmod{p}$
7: **if** $r = f(t)$ **then**
8: **return** Verifica
9: **else**
10: **return** No Verifica
11: **end if**

Para ECDSA [NIST, 2013] los parámetros de dominio están dados por la tupla (H, K, E, q, G), donde H es una función picadillo, E es una curva elíptica definida sobre el campo finito K y G es un punto en la curva con orden primo q. Por lo que de nuevo los parámetros de dominio definen una función picadillo, un grupo de orden q y un generador del grupo. Para facilitar la comprensión los puntos en la curva elíptica serán denotados con letra mayúscula. Con los parámetros de dominio también se incluye un entero h, llamado el cofactor, tal que

$$\#E(K) = h \cdot q.$$

Esto es debido a que el valor de h es importante en otros protocolos.

Usualmente se elige una curva en la cual $h \leq 4$. El par de llaves pública/privada está dado por (Y, x) donde

$$Y = xG$$

y la función f está definida de la siguiente forma

$$f : \begin{cases} E & \to \mathbb{F}_q \\ P & \mapsto x(P) \pmod{q} \end{cases}$$

Donde $x(P)$ denota la coordenada x del punto P la cual se interpreta como un número entero al momento de realizar la reducción módulo q. Esto se realiza aun cuando la curva está definida sobre un campo con característica dos. En el caso de campos con característica par, se requiere una convención de cómo convertir un elemento en el campo, los cuales son normalmente un polinomio $g(x)$, a un entero. Casi todos los estándares adoptaron la convención de evaluar $g(2)$ en los enteros. Por lo que el polinomio

$$x^5 + x^2 + 1$$

se interpreta como el número entero 37, debido que

$$37 = 32 + 4 + 1 = 2^5 + 2^2 + 1$$

Los algoritmos de ECDSA se derivan de los correspondientes a DSA de la siguiente manera:

Algoritmo Firma ECDSA

Entrada: Un mensaje m y la llave privada x.
Salida: Una firma (r, s) del mensaje m.

1: Elegir $k \in \mathbb{R}\{1, \ldots, q-1\}$
2: $T \leftarrow kG$
3: $r = f(T)$
4: **if** $r = 0$ **then** `goto` 1
5: **end if**
6: $e \leftarrow H(m)$
7: $s \leftarrow (e + xr)/k \pmod{q}$
8: **if** $s = 0$ **then** `goto` 1
9: **end if**
10: **return** (r, s)

Algoritmo Verificación de Firma ECDSA

Entrada: Un mensaje m , una firma (r, s) y la llave pública Y.
Salida: Verifica / No Verifica.

1: **if** $r, s \notin \{1, \ldots, q-1\}$ **then**
2: **return** No Verifica
3: **end if**
4: $e \leftarrow H(m)$
5: $u_1 \leftarrow e/s \pmod{q}, u_2 \leftarrow r/s \pmod{q}$
6: $T \leftarrow u_1 G + u_2 Y$
7: **if** $r = f(T)$ **then**
8: **return** Verifica
9: **else**
10: **return** No Verifica
11: **end if**

Un aspecto importante tanto de DSA como de ECDSA es que las llaves efímeras k requieren ser verdaderamente aleatorias. El porqué de este requerimiento se puede apreciar en el siguiente ejemplo:

Considere el caso que alguien firma dos mensajes diferentes m y m', con el mismo valor de k. Las firmas son entonces (r, s) y (r', s') donde

$$\begin{gathered} r = r' = f(kG) \\ s = (e + xr)/k \pmod{q}, \text{ donde } e = H(m) \\ s' = (e' + xr)/k \pmod{q}, \text{ donde } e' = H(m'). \end{gathered}$$

Entonces tenemos que

$$(e + xr)/s = k = (e' + xr)/s' \pmod{q}$$

de donde podemos deducir

$$xr(s' - s) = se' - s'e$$

y por lo tanto

$$x = \frac{se' - s'e}{r(s' - s)} \pmod{q}$$

logrando de esta manera obtener la llave privada x.

CAPÍTULO 4

PKI

The trust of the innocent is the liar's most useful tool.

—STEPHEN KING

La habilidad de crear, manipular y compartir documentos digitales ha creado un cúmulo de nuevas aplicaciones, pero también ha dado pie a nuevos problemas, como son el proteger la confidencialidad e integridad de los documentos al ser almacenados y transmitidos. Con el uso de criptografía de llave pública se tiene una solución a estos, debido a que no es necesario establecer una llave común para cifrar datos y a la vez se tiene la capacidad de firmar datos, asegurando de esta manera la privacidad y confidencialidad. Aunque estas primitivas criptográficas sean en concepto fáciles de lograr, en el mundo real recaen en la habilidad de poder asociar una llave pública a una identidad acorde a la operación (un nombre, un correo electrónico, un identificador legal, etc.). La infraestructura de llave pública (*PKI* por sus siglas en inglés) es el término empleado para referirse a los protocolos y elementos necesarios para realizar esta asociación. [Vacca, 2013]

El objetivo principal del desarrollo de la infraestructura de llave pública es hacer posible la obtención de llaves públicas de manera segura, conveniente y eficiente. [Stallings, 2013]

El RFC 4949 [Shirey, 2007] (Internet Security Glossary) define la infraestructura de llave pública como:

> El conjunto de hardware, software, personas, políticas y procedimientos necesarios para crear, administrar, almacenar, distribuir y revocar certificados digitales basados en criptografía asimétrica.

4.1. Origen de PKI

La historia de PKI se remonta hasta el artículo seminal de Whitfield Diffie y Martin Hellman de 1976 sobre criptografía de llave pública [Diffie and Hellman, 1976] en el cual se lee:

> Dado un sistema de este tipo, el problema de la distribución de llaves se simplifica de forma significativa. Cada usuario genera un par de transformaciones inversas E y D en su terminal. La transformación de descifrado D debe ser conservada de forma secreta y nunca debe comunicarse por canal alguno. La llave de cifrado E puede ser dada a conocer, colocándola en algún directorio público junto con el nombre y dirección del usuario. Cualquiera puede cifrar mensajes y enviarlos al usuario, pero nadie puede descifrar los mensajes destinados a él. Por esto, los criptosistemas de llave pública pueden ser considerados como *cifrados de acceso múltiple*.
> Es crucial que el directorio público con las llaves de cifrado esté protegido contra modificaciones no autorizadas. Esta tarea se facilita gracias a la naturaleza pública del archivo. Protegerlo contra lectura es innecesario y, debido a que el archivo es modificado de forma infrecuente, se pueden emplear mecanismos elaborados de protección contra escritura de forma económica.

Viendo algunas de las desventajas de esta propuesta —como lo son: el posible cuello de botella, el directorio se vuelve un objetivo muy valioso y que el hecho de inhabilitar el directorio restringe a la vez la capacidad de los usuarios en comunicarse de forma segura— Loren Kohnfelder propuso el concepto de certificados en 1978 [Kohnfelder, 1978].

4.2. Certificados

Supongamos que Alicia quiere comunicarse con Beto. Para ello Alicia obtiene la llave pública de Beto de un servicio de directorio. Alicia debe de estar segura de que la llave sea verdaderamente de Beto, en el caso de que esto no sea posible ella podría sufrir un ataque del intruso de en medio.

En la figura 4.1 se muestra un esquema del posible ataque: Alicia recibe una llave pública k_{p1} para la cual Eva posee la correspondiente llave privada k_{s1}. Alicia creyendo que ha recibido la llave pública de Beto cifra el mensaje que desea enviar y lo manda. Eva intercepta el mensaje, lo descifra, lo lee y finalmente lo cifra con la verdadera llave pública de Beto k_{p2} y lo envía. Beto recibe el mensaje lo descifra y lo lee. Tanto Alicia como Beto creen que el mensaje se ha mantenido secreto. Eva podría emplear la misma estrategia en el sentido inverso de la comunicación y enterarse de la respuesta de Beto.

Para evitar esta situación se emplean los certificados. Los certificados son estructuras de datos las cuales asocian llaves públicas con una identidad, un tercero de confianza firma la información contenida *certificando* que la información ha sido verificada y es verídica. Los certificados separan las funciones de búsqueda y autenticidad al permitir a una autoridad certificadora asociar un nombre con una llave pública a través de una firma digital, y después almacenar el certificado en un repositorio. Ya que el repositorio no tiene que ser autenticado y se puede replicar, volviéndolo así tolerante a fallas y a ataques de negación de servicio, el enfoque de las autoridades certificadoras elimina muchos de los problemas asociados a los directorios autenticados.

Si Alicia posee un certificado de la llave pública de Beto y además confía en la llave de verificación de firma del emisor, entonces al verificar la firma del certificado Alicia se convence de la autenticidad de la llave pública de Beto. De esta manera los certificados trasladan la confianza en una llave pública a la confianza en

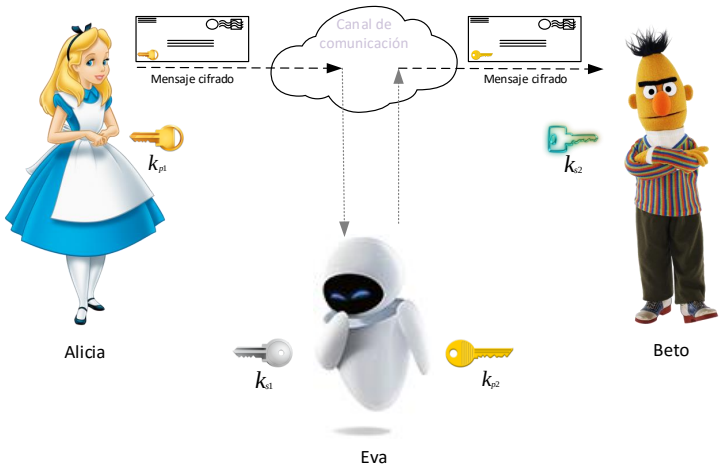


Figura 4.1: Ataque del intruso de en medio

una autoridad.

El contenido mínimo de un certificado es:

⇛ El nombre del *sujeto* con quien la llave pública del certificado está asociada.

⇛ La *llave pública* que está asociada a la entidad.

⇛ El *algoritmo criptográfico* con el cual se empleará la llave pública.

⇛ El *número de serie* del certificado.

⇛ El *período de vigencia* del certificado.

⇛ El nombre del *emisor* que firmó el certificado.

⇛ *Restricciones* aplicables al uso de la llave pública en el certificado.

Típicamente el emisor de un certificado es diferente al sujeto. Aunque, también existen los certificados *autoemitidos* en los cuales el emisor y el sujeto son el mismo. Estos son empleados, por ejemplo cuando un emisor realiza un cambio de políticas. Un caso especial de los certificados autoemitidos son los certificados *autofirmados*, los cuales tienen la característica adicional de que la llave pública firmada es la misma que se emplea para verificar la firma del certificado. Los certificados autofirmados se emplean para publicar las llaves públicas de los emisores. [Buchmann et al., 2013]

4.3. Modelos de confianza

La criptografía de llave pública sólo puede ser usada en la práctica si los usuarios confían en la autenticidad de las llaves públicas.

A continuación se presentan los principales modelos de confianza [Buchmann et al., 2013].

4.3.1. Confianza directa

Es el modelo de confianza más básico. De hecho, todos los demás modelos requieren de él para establecer inicialmente la confianza. En este modelo la llave es obtenida directamente del propietario o el propietario confirma de forma directa y convincente la autenticidad de la llave.

Los sistemas operativos, clientes de correo y navegadores de Internet emplean este modelo al suministrar precargados varios certificados autofirmados de emisores en que sus creadores confían.

4.3.2. Red de confianza

La confianza directa no es suficiente para lograr que la criptografía asimétrica sea útil en la práctica. Por ejemplo, si un usuario quiere enviar un correo electrónico cifrado a otro usuario, el establecer confianza directa requiere de que el emisor previamente contacte al receptor para obtener su llave pública. Aunque esto es posible, la comunicación se vuelve considerablemente menos ágil. Debido a esto otros modelos han sido propuestos.

Una alternativa es el modelo de red de confianza, inventado por Phill Zimmermann en 1992 para su sistema PGP [Zimmermann, 1995]. El modelo está inspirado en redes sociales de confianza. En el los usuarios confían en una llave si es obtenida directamente de su dueño (confianza directa) o si un número suficiente de usuarios conocidos y confiables recomiendan confiar en ella. La idea básica de PGP es: los usuarios que confían en una llave pública la firman. El funcionamiento de la versión estandarizada de PGP —OpenPGP— se detalla en el RFC 4880 [Callas et al., 2007].

4.3.3. Confianza jerárquica

El modelo de red de confianza es bastante conveniente, escalable y no requiere de infraestructura centralizada. Pero su rango de aplicación es limitado. Normalmente quien ha firmado una llave pública no acepta ningún tipo de responsabilidad legal respecto a la autenticidad de la llave pública que firmó. Debido a esto el modelo de red de confianza no es adecuado para la realización de operaciones financieras y de negocios.

En el modelo de confianza jerárquica los emisores aceptan la responsabilidad de los certificados que ellos firman. También en una PKI jerárquica la confianza en las llaves públicas depende de la confianza en un firmante determinado, llamado *ancla de confianza* (*trust anchor* en inglés).

En una PKI jerárquica las llaves públicas son certificadas por una *autoridad certificadora* (*CA* por sus siglas en inglés). Cada CA implementa un proceso de verificación de autenticidad para las llaves públicas que certifica. Este proceso es publicado para que los usuarios puedan determinar su nivel de confianza en la CA.

En la figura 4.2 se presenta el esquema de una PKI jerárquica muy simple. Las entidades participantes se representan con un rectángulo y cada línea representa un certificado. En esta PKI sólo existe una CA que ha emitido certificados a las entidades finales Alicia, Beto y Carlos. La CA es el ancla de confianza y ha emitido un certificado autofirmado a ella misma. Este certificado contiene la llave pública que se debe de usar para verificar las firmas de los certificados emitidos por esta CA. Todas las entidades en la PKI establecen confianza directa en el ancla de confianza. Debido a que los usuarios de la PKI confían en el ancla de con-

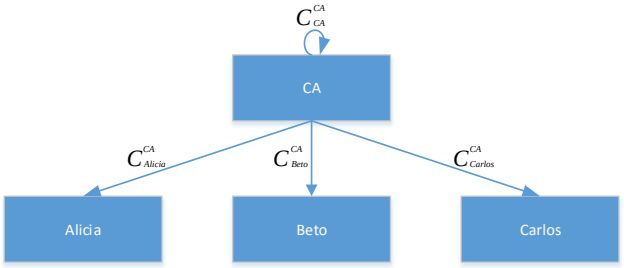


Figura 4.2: Una PKI jerárquica

fianza, los usuarios de la PKI confían en la autenticidad de las llaves públicas de Alicia, Beto y Carlos. De igual manera, si alguna entidad fuera de la PKI confia en el ancla de confianza entonces también aceptará las llaves públicas de Alicia, Beto y Carlos.

En general una PKI jerárquica tiene la estructura de un árbol como se muestra en la figura 4.3. Los nodos internos son CAs, las hojas son entidades finales y la raíz del árbol es llamada la *autoridad certificadora raíz*. Esta última es el ancla de confianza. Cada flecha de A a B representa un certificado emitido a B por la CA A.

En una PKI jerárquica la confianza en la autenticidad de las llaves públicas queda establecida a través de un *camino de certificación*. Este camino es una secuencia finita de certificados en la cual el sujeto es el emisor del siguiente certificado, la única excepción es el último certificado. El camino establece confianza en la llave pública de una entidad si:

- El sujeto en el último certificado del camino es una entidad final.
- El emisor en el primer certificado es el ancla de confianza.

Por ejemplo, empleando la PKI de la figura 4.3 Alicia confía en la llave pública de Diana. Para establecer esta confianza ella utiliza el camino de certificación $C^{CAR}_{CAR}, C^{CAR}_{CA2}, C^{CA2}_{Diana}$.

El camino empieza con el certificado autofirmado de la CA raíz. En el siguiente certificado la CA raíz certifica la llave pública de la CA2. Finalmente en el tercer certificado la CA2 certifica la llave pública de Diana.

Desde el punto de vista de Diana, ella confía en la llave pública de Alicia ya que establece confianza utilizando el camino de certificación $C^{CAR}_{CAR}, C^{CAR}_{CA1}, C^{CA1}_{CA3}, C^{CA3}_{Alicia}$.

Extendiendo el camino de certificación

En la práctica pueden existir muchas PKI jerárquicas independientes, las cuales son administradas por entidades diversas (gobiernos, universidades, compañías, etc.). Los participantes de cada PKI son capaces de autenticar las llaves públicas de su misma PKI. Pero los usuarios de PKIs diferentes no pueden autenticar sus respectivas llaves públicas a menos que sus PKIs se unan.

A continuación se describen varios métodos para combinar PKIs.

Listas de confianza

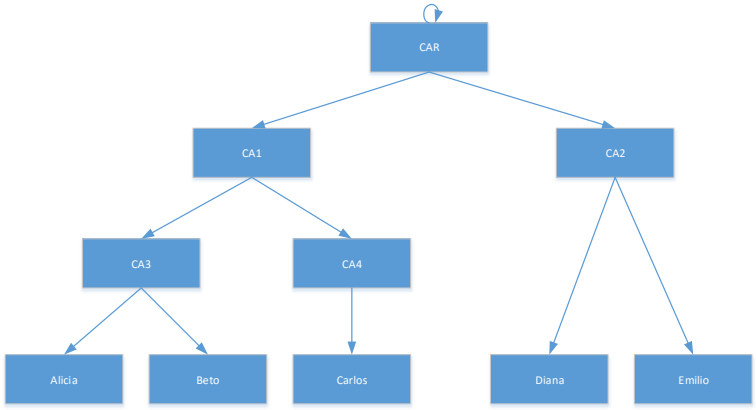


Figura 4.3: La jerarquía de una CA

Una forma sencilla de lograr que una entidad pueda autenticar las llaves públicas de entidades finales en diferentes PKIs jerárquicas es: aceptar el ancla de confianza de esas PKIs como de confianza. Para esto no se requiere ser miembro de la PKI. Por ejemplo, en la figura 4.4 Alicia puede aceptar a la CA2 como una nueva ancla de confianza. Así ella logra construir un camino de certificación que involucra a la CA2. Con esto Alicia estaría confiando en las llaves públicas de Emilia, Francisco y Gregorio, aunque ella no forme parte de esa PKI. Una entidad que quiere emplear llaves públicas de diferentes PKIs jerárquicas mantiene una *lista de confianza* de las CA raíz. Esto se realiza en los navegadores web y en los sistemas operativos, se incluye una lista de CAs de confianza la cual los usuarios pueden administrar.

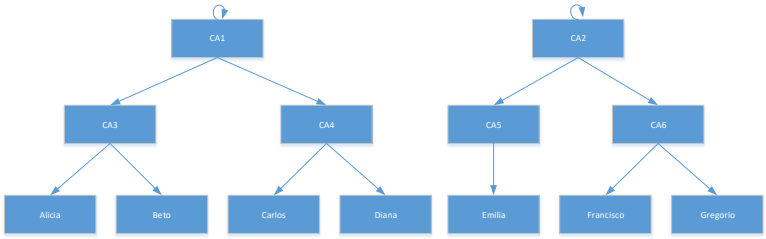


Figura 4.4: Dos PKIs independientes

Raíz común

Otra forma de unir dos o más CAs jerárquicas es agregar un nivel en el camino de certificación con una nueva CA *raíz común* como se muestra en la figura 4.5. Cada entidad final debe de remplazar su ancla de confianza por la nueva raíz común.

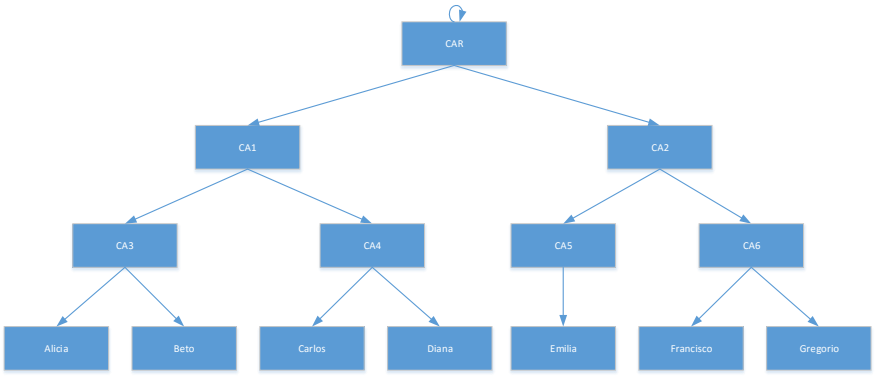


Figura 4.5: Dos PKIs unidas por una raíz común

Certificación cruzada

La *certificación cruzada* permite a los usuarios de dos o más PKIs diferentes poder validar las llaves públicas de los demás sin tener que remplazar su ancla de confianza. La idea es que dos CAs certifiquen la llave pública de la otra CA. El concepto se ejemplifica en la figura 4.6, donde podemos ver que la CA1 emitió un certificado a la CA2 y viceversa. Por ejemplo, Alicia autentica la llave pública de Emilia empleando el camino de certificación $C^{CA1}_{CA1}, C^{CA1}_{CA2}, C^{CA2}_{CA5}, C^{CA5}_{Emilia}$.

El principal problema que se tiene al emplear certificación cruzada es el número de certificados que se requiere crear para conectar varias PKIs. Para conectar n PKIs se requiere emitir $n(n-1)$ certificados cruzados, lo cual puede ser impráctico.

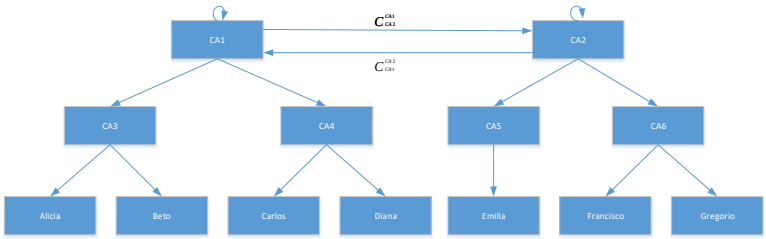


Figura 4.6: Dos CAs empleando certificación cruzada

CAs Puente

El emplear *CAs puente* permite a los usuarios de varias PKIs el poder autenticar las llaves públicas de los demás sin tener que remplazar su ancla de confianza. También comparado con certificación cruzada requiere de menos certificados. Para lograr esto se crea una nueva CA para la cual las CAs raíz que se van a integrar emplean certificación cruzada. En la figura 4.7 podemos ver el siguiente caso: cuando Alicia, quien tiene

como ancla de confianza a la CA1, quiere autenticar la llave pública de Francisco, cuya ancla de confianza es la CA2. El camino de certificación sería: $C^{CA1}_{CA1}, C^{CA1}_{CAPuente}, C^{CAPuente}_{CA2}, C^{CA2}_{CA6}, C^{CA6}_{Francisco}$. Se puede ver que en este camino no es necesario que Alicia cambie su ancla de confianza.

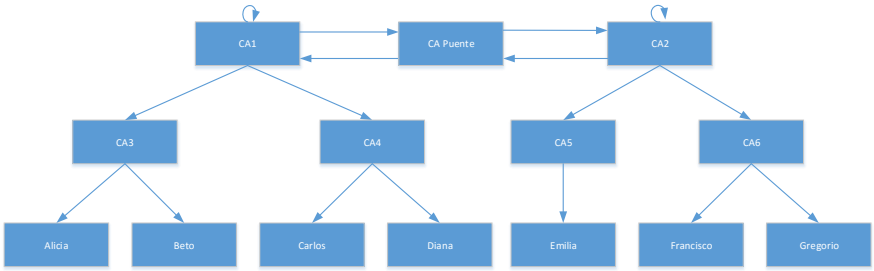


Figura 4.7: Una CA puente uniendo dos PKIs

4.4. PKI en la práctica

En la práctica es importante aclarar para que sí sirve PKI y para que no sirve, además de para quién es útil. También es necesario saber cómo se integra con las tecnologías complementarias que se emplean para proveer seguridad a un ambiente.

Los malentendidos con respecto al rol de una PKI son frecuentes y crean ya sea muy altas o muy bajas expectativas así como una subsecuente implementación y uso deficiente. En tales situaciones la función de una PKI dentro del horizonte completo de seguridad de la información no se puede apreciar y por lo tanto su valor no se puede percibir ni explotar [Adams and Lloyd, 2002].

4.4.1. Para que SÍ sirve PKI

PKI es una tecnología de autenticación, una forma para identificar entidades en un ambiente. La criptografía de llave pública es empleada en conjunto con los siguientes componentes para crear una tecnología que permita la identificación de entidades:

- Un mecanismo para establecer confianza de acuerdo a un modelo de confianza definido.
- Un mecanismo para nombrar a las entidades de tal forma que cada entidad pueda ser identificada de forma única en el ambiente de interés.
- Un mecanismo para distribuir información respecto a la validez de la asociación entre una llave pública y un nombre a otras entidades interesadas dentro del ambiente.

PKI provee autenticación —ni más, ni menos—. El saber quién es alguien no necesariamente implica que se sepa qué privilegios posee, que se pueda confiar automáticamente en la entidad, ni que se tenga la información necesaria para realizar algún paso en una transacción. Muchos pasos en algunas transacciones requieren información de autorización, privilegios o atribución acerca de una entidad; PKI no entra en estos roles, lo único que puede hacer es proporcionar integridad, autenticidad o confidencialidad para esta información. En su forma más pura PKI provee de manera confiable un nombre, sin garantizar nada más.

4.4.2. Para que NO sirve PKI

En primer lugar, como ya se mencionó, PKI típicamente no proporciona autorización. Aunque es posible incorporar estos datos en la PKI (dentro del certificado, por ejemplo), esto no es una práctica común ni es recomendado.

Tampoco PKI proporciona confianza en las demás entidades finales. PKI inicia con el establecimiento de un mecanismo de confianza cuyo único propósito es que cada entidad confíe en una CA. A partir de este punto PKI permite a una entidad confiar en otra llave a partir del hecho de que la entidad cree que esa llave verdaderamente pertenece a una entidad con el nombre especificado. Si Alicia puede llegar a confiar en que una llave pública realmente le pertenece a Beto, esto le puede dar cierta seguridad, pero esto es totalmente diferente a que ella confíe en Beto. Alicia debe de emplear otro tipo de información para tomar esta decisión. PKI no hace que Alicia confíe en Beto, ella empieza por confiar en su CA y después emplea PKI para convencerse de que una llave dada le pertenece a una entidad llamada “Beto”.

En tercer lugar, PKI no crea nombres únicos para las entidades, no está entre sus metas el resolver el problema de cómo nombrar a las entidades. El trabajo de PKI es unir llaves públicas con nombres. El nombre corresponde a alguna entidad del mundo real como lo son un humano, una máquina o un programa. Esa entidad del mundo real debe poder ser identificada de manera única dentro de su ambiente o no tendría sentido el asociarla con un mecanismo de autenticación.

PKI de ningún modo hace que las aplicaciones, sistemas operativos y plataformas de cómputo sean “seguras”. Un mecanismo de autenticación no puede en general compensar errores o malas prácticas de programación en las aplicaciones y sistemas operativos. No puede prevenir desbordamientos de buffer, ataques de denegación de servicio, virus, gusanos o troyanos en una red. PKI no libera a los administradores a implementar cortafuegos, detectores de intrusos, actualizar los sistemas, realizar respaldos, etc., todos estos deben de implementarse en un sistema que se quiera proteger ya sea que se use o no PKI. Una autenticación confiable no hace por sí misma a un sistema más seguro, es una de un conjunto de herramientas que deben de ser usadas para robustecer la seguridad de un ambiente.

Finalmente, PKI no hace el comportamiento de los usuarios humanos (finales o administradores) más confiable o correcto. Un usuario final seguirá escribiendo su contraseña en un papel pegado a su computadora, un administrador seguirá realizando configuraciones erróneas, un administrador malintencionado seguirá siendo capaz de comprometer el ambiente. El entrenamiento de los usuarios es necesario en cualquier ambiente, ya sea que emplee PKI u otro método de autenticación.

4.5. X.509

Existen varios estándares que especifican el formato que deben tener los certificados. El estándar X.509 [ITU, 2012b] es el más usado. Su formato es empleado en los protocolos SSL/TLS para proteger las comunicaciones por Internet y en el estándar S/MIME para correo electrónico seguro.

El grupo de trabajo de Infraestructura de Llave Pública X.509 de la organización internacional Internet Engineering Task Force (IETF) ha sido el promotor para realizar un modelo formal (y genérico) basado en X.509 apto para la implementación en Internet de una arquitectura basada en certificados [Stallings, 2013].

El modelo de X.509 es el estándar prevaleciente para PKIs basadas en certificados. El estándar ha evolucionado de tal forma que las aplicaciones en Internet que requieren de PKI se basan prácticamente en los lineamientos de la IETF los cuales han desarrollado y extendido las ideas de X.509. Los certificados de tipo X.509 son empleados como base en SSL, TLS, varias implementaciones de red privada virual, PKIs guber-

namentales y muchos sistemas ampliamente empleados [Vacca, 2013].

4.5.1. Historia

Algunos años después de que Loren Kohnfelder inventara el concepto de certificado [Kohnfelder, 1978], éste fue incorporado a X.500, un directorio global impulsado por un conglomerado de compañías de telecomunicaciones. El modelo propuesto para el directorio de X.500 era una base de datos jerárquica, con rutas a través del directorio definidas por una serie de nombres distinguidos relativos (RDN) que agrupados forman un nombre distinguido (DN). Para proteger el acceso al directorio, los diseñadores implementaron diversos mecanismos que van desde el uso de contraseñas hasta el relativamente novedoso (en ese entonces) uso de firmas digitales. Cada porción del directorio tenía CAs ligadas que emitían certificados para el control de acceso (ver figura 4.8). La estructura original de los certificados X.509 v1 tenía un campo para el DN del emisor y otro para el DN del sujeto con el fin de poder colocar los certificados en el directorio.

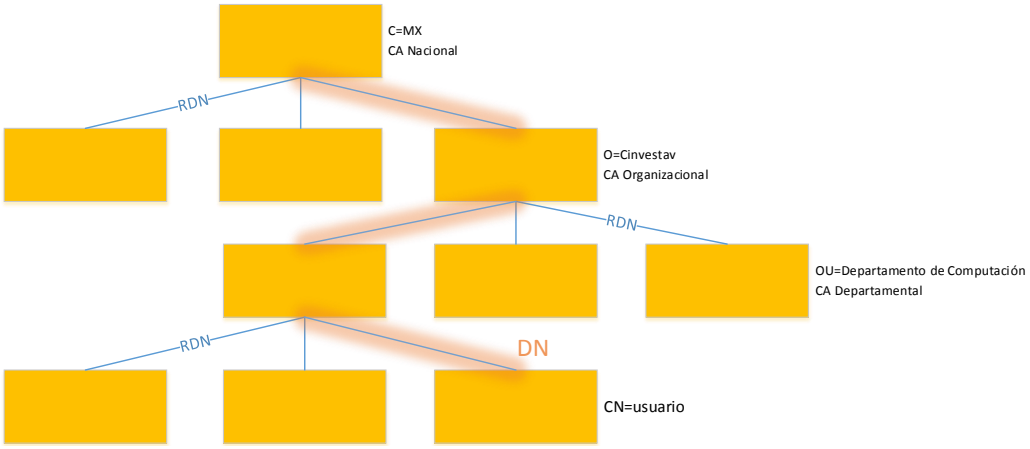


Figura 4.8: Modelo de directorios y certificación de X.500

X.509 es parte del estándar de directorio X.500 [ITU, 2012a] creado por la Unión Internacional de Telecomunicaciones (ITU). X.500 especifica un directorio jerárquico útil para el conjunto de mensajes del estándar X.400 [ITU, 1999]. Como tal éste posee un sistema de nombres que describe a las entidades según su posición en una jerarquía. Un ejemplo de cómo se ve un nombre en X.500/X.400 es:

CN=Juan Pérez, OU=Computación, O=Cinvestav, C=MX

Este nombre describe a una persona con el *nombre común* (CN) "Juan Pérez" que trabaja en la *unidad organizacional* (OU) llamada "Computación", dentro de la *organización* (O) llamada "Cinvestav" en México. Los componentes del nombre fueron diseñados para ser ejecutados por su respectivo componente del directorio — habría directorios de países que apuntarían a directorios de organizaciones y así sucesivamente— y esta descripción jerárquica fue plasmada en el diseño de X.509. Muchos de los cambios realizados por el IETF y otros organismos han adaptado el modelo jerárquico a la naturaleza más distribuida de Internet.

4.5.2. Modelo de certificados

El modelo de X.509 especifica un sistema de autoridades certificadoras las cuales emiten certificados a las entidades finales (usuarios, sitios web y entidades que poseen llaves privadas). Un certificado emitido por una CA contiene (entre otros datos) el nombre de la entidad final, el nombre de la CA, la llave pública de la entidad final, el periodo de vigencia y el número de serie del certificado. Para validar un certificado la entidad receptora emplea la llave pública de la CA para verificar la firma en el certificado, verifica que la fecha esté dentro del periodo de vigencia y opcionalmente realiza algunas validaciones en línea.

Para que este proceso pueda llevarse a cabo se requiere contar con la llave pública de la CA, ésta se obtiene de otro certificado que certifica la llave pública de la CA.

Otro elemento que se requiere para que el sistema pueda funcionar de forma segura, es que la CA debe de tener la capacidad de deshacer una acción de certificación. Aunque un certificado asocia una identidad con una llave pública existen situaciones en las que este vínculo se debe deshacer, por ejemplo, cuando un trabajador deja de laborar en una empresa es necesario impedir que pueda seguir firmando como empleado. También es necesario invalidar el certificado en caso de pérdida o compromiso de la llave privada. Esto se logra mediante un protocolo de validación donde el usuario que examina el certificado pregunta a la CA si éste sigue siendo válido.

4.5.3. Arquitectura de implementación

Aunque en teoría la CA es la entidad encargada de crear y validar los certificados, en la práctica hay veces que es deseable o necesario delegar las actividades de autenticación y certificación de los usuarios a otros servidores. La seguridad de la llave privada de la CA es crucial para todo el sistema de PKI. Al limitar las funciones del servidor que requieren de esta llave se disminuye el riesgo de compromiso o uso indebido. La arquitectura de X.509 define un rol de servidor delegado, la *autoridad registradora* (RA), la cuál permite delegar la autenticación. Extensiones posteriores a la arquitectura X.509 han creado otro rol delegado, la *autoridad de validación* (VA), quien tiene la tarea de contestar las peticiones acerca de la validez de un certificado.

Una autoridad registradora es típicamente usada para distribuir la función de autenticación al emitir los certificados sin tener que distribuir la llave privada de la CA. La función de la RA es realizar la autenticación necesaria de la entidad final para poder emitir un certificado, después envía un acuse firmado de haber realizado la verificación tanto de la entidad como de la llave pública a ser certificada. La CA valida el mensaje de la RA y emite un certificado.

Las autoridades de validación tienen la habilidad de revocar certificados, liberando a la CA de esta función. A través de una cuidadosa implementación de RAs y VAs es posible construir una arquitectura de certificados donde el servidor critico de la CA sólo es accesible a un número limitado de servidores y junto con la correcta implementación de medidas de seguridad de red es posible reducir o hasta eliminar riesgos que provengan fuera de la red de la PKI.

4.5.4. Validación de certificados

La validación de certificados X.509 es un proceso complejo y se puede realizar con diferentes niveles de confianza. Dependiendo del tipo de aplicación es el nivel de verificación que debe de llevarse a cabo. A continuación se menciona el proceso típico de validación de un certificado X.509.

El proceso típico de validación de un certificado involucra tres pasos y requiere de tres entradas. La primera es el certificado a ser validado, la segunda son los certificados intermedios y finalmente se requiere del almacén de certificados intermedios y raíz en que la aplicación confía. Los pasos descritos a continuación

son una descripción simplificada ya que en la práctica el uso de CAs puente y demás modelos extendidos de confianza llevan a una verificación mucho más compleja. La IETF en su RFC 3280 [Housley et al., 2002] presenta una especificación completa para la validación de certificados.

Paso 1: Validación de firmas y construcción del camino de certificación

No se puede confiar en el contenido del certificado hasta que se haya verificado la firma. Por lo tanto, la primera acción es verificar la firma del certificado. Para hacerlo se debe de localizar el certificado de la autoridad que firmó el certificado a verificar. Esto se logra buscando en los certificados intermedios y en el almacén local de certificados alguno cuyo sujeto sea el emisor del certificado a validar. Si existen varios certificados que cumplan esa propiedad entonces el validador puede emplear un campo extendido llamado *identificador de llave del sujeto* dentro del conjunto de posibles candidatos. Si aún se tienen varios candidatos entonces se emplea el que tenga la fecha de expedición más reciente. Ya que se ha encontrado el certificado de la autoridad se verifica la firma del certificado. Si la firma no verifica se detiene el proceso de validación y se rechaza el certificado.

Si la firma verifica y se confía en el certificado de la autoridad entonces el camino de certificación se transfiere al siguiente paso. Si no se confiá en la autoridad entonces el certificado de la autoridad es tratado como un certificado a validar y el paso uno es ejecutado recursivamente hasta que se construya un camino a un certificado de confianza o exista un error.

Para que se pueda construir un camino de certificación es necesario que el validador posea todos los certificados en el camino. Para esto se requiere que el validador posea un almacén de certificados intermedios o que el protocolo que emplea el certificado proporcione todos los certificados intermedios necesarios.

Paso 2: Validación de fechas, políticas y uso de llave

Ya que se tiene un camino de certificación es necesario verificar varios campos para asegurar que el certificado fue emitido correctamente y es válido. Las siguientes verificaciones se deben de realizar en el camino de certificación:

Las fechas de todo el camino son correctas. Cada certificado posee un periodo de vigencia, con una fecha de inicio y fin. Para aplicaciones diferentes a la verificación de una firma en un documento la fecha debe de encontrarse en el periodo de vigencia. Algunas aplicaciones requieren de validación anidada de tiempo, lo que quiere decir que el certificado es aceptado si y sólo si el certificado del emisor es también válido. Depende de las políticas definidas para la aplicación si un certificado fuera del periodo de vigencia es rechazado o si sólo una alerta es presentada al usuario. También las aplicaciones pueden tratar de manera diferente un certificado que aún no es efectivo a uno que ha expirado.

Las aplicaciones que validan un certificado en un documento almacenado pueden revisar la fecha en que se firmó el documento y no la fecha en que se verifica la firma. Existen tres casos particulares de interés. El primero es cuando la firma de un documento es válida y todos los certificados en el camino de certificación se encuentran en su periodo de vigencia. En este caso todas las fechas son válidas y la verificación continúa con los demás pasos. El segundo caso es cuando el camino de certificación se encuentra fuera del periodo de vigencia ya sea por uno o varios certificados y se cree que el documento fue firmado cuando el camino ya no era válido. En este caso las fechas están fuera del periodo de vigencia y como mínimo se debe de alertar al usuario. Finalmente se llega al caso ambiguo donde el camino de confianza se encuentra fuera de vigencia, pero se cree que cuando se firmó el documento este era válido. Dependiendo de la política implementada por la aplicación este caso puede ser tratado de diferentes maneras. Se pueden tratar de forma estricta las fechas y la validación se considera fallida. De forma alternativa la aplicación podría considerar que los certificados estaban vigentes cuando se realizó la firma y se continuaría con los demás pasos de validación.

La aplicación también podría implementar diferentes medidas para que este caso no se presente, como emplear un servicio de marcado de tiempo cuando se firman los documentos o auxiliarse de un mecanismo en el que los documentos sean refirmados antes que el camino de certificación expire.

Una vez que el camino de certificación ha sido construido el verificador también debe de revisar varios campos extendidos de X.509. Algunos de los campos importantes en la validación del camino de confianza son:

- **BasicConstraints**: Este campo se emplea por las CAs y limita el tamaño de la cadena de confianza después del certificado de la CA.
- **NameConstraints**: Este campo limita el espacio de nombres certificados en la cadena después del certificado de la CA. Esto puede ser usado para limitar a una CA específica la emisión de certificados para un domino o un espacio de nombres de X.400.
- **KeyUsage** y **ExtendedKeyUsage**: En estos campos se limita el uso que se puede dar a una llave certificada.

Paso 3: Consultar autoridades de revocación

Finalizada la verificación de la cadena de confianza, fechas y uso de llaves se puede consultar a las autoridades de revocación que se listan en cada certificado para verificar que todos los certificados sigan siendo válidos. Los certificados pueden contener campos extendidos los cuales indican la dirección de la *lista de revocación de certificados* o el nombre de servidores de protocolo en línea de estado de certificados. Con estos métodos se puede verificar que los certificados no hayan sido revocados.

4.5.5. Revocación

Debido a que los certificados son válidos por un amplio periodo de tiempo, es posible que durante el período de vigencia la llave privada sea comprometida, una identidad cambie u ocurra un evento que invalide la asociación de identidad con la llave pública. Para estos casos, la CA debe de poder revocar un certificado. Para que las aplicaciones validen el estado de un certificado existen dos mecanismos: listas de revocación de certificados y el protocolo en línea de estado de certificados.

CRLs

La arquitectura original de X.509 implementaba el proceso de revocación a través de listas de revocación de certificados (CRLs). Una CRL es un documento expedido periódicamente el cual contiene una lista de los números de serie de los certificados que han sido revocados por la CA. Cuando el certificado de una CA es revocado por una CA superior en la jerarquía de la PKI se emite una lista de revocación de autoridad (ARL), la cual tiene el mismo formato que una CRL. Las CRLs y ARLs como se definen en X.509 y el RFC 3280 son objetos codificados en ASN.1 [ITU, 2015] que contienen los datos mostrados en la tabla 4.1 como encabezado.

Después del encabezado se encuentran los datos de los certificados que han sido revocados. Para cada certificado se incluyen los datos mostrados en la tabla 4.2.

Al verificar la CRL se debe constatar que esté firmada con la llave del emisor indicado en el encabezado y que la fecha actual se encuentre entre las indicadas en los campos *thisUpdate* y *nextUpdate*. Esta acción es crucial ya que de no ser realizada un atacante podría emplear un certificado revocado al proveer una CRL previa donde no se incluya este. Los certificados que ya han expirado son removidos de la lista para evitar el crecimiento desmedido de la CRL.

Versión	Especifica el formato de la CRL. Actualmente es empleada la versión 2.
Algoritmo de firma	Especifica el algoritmo que se empleó para firmar la CRL.
Emisor	Nombre de la CA que emitió la CRL
thisUpdate	Fecha a partir de la cual la CRL es válida.
nextUpdate	Fecha cuando la siguiente CRL será emitida.

Tabla 4.1: Encabezado de una CRL de X.509

Número de Serie	El número de serie del certificado revocado.
Fecha de revocación	Fecha a partir de la cual la revocación es efectiva.
Datos adicionales	(opcional) Especifica el porqué fue revocado.

Tabla 4.2: Datos de cada certificado revocado en una CRL de X.509

CRL delta

En sistemas muy grandes los cuales emiten una gran cantidad de certificados las CRLs pueden volverse muy extensas. Una estrategia para prevenir la posible saturación de la red debido a la transferencia de la CRL completa a cada solicitante es el emitir CRLs delta junto con una CLR base. La CRL base contiene los certificados revocados hasta cierto punto en el tiempo y las CRLs delta contienen únicamente los certificados que se han agregado a la lista base en un periodo de tiempo. Así los clientes deben descargar la lista base con menos frecuencia y sólo deben obtener listas delta más pequeñas para saber cuales certificados han sido revocados recientemente. Las listas delta son prácticamente iguales a las CRLs normales sólo incluyen un campo adicional que indica que son de tipo delta.

OCSP

El protocolo en línea de estado de certificados u *online certificate status protocol* (OCSP) por su nombre en inglés fue diseñado con la meta de reducir el costo asociado a la transmisión de las CRLs así como para eliminar la ventana de tiempo entre la invalidación y revocación de un certificado asociada a la periodicidad de las CRLs. La idea detrás de OCSP es muy simple. Cada certificado emitido por la CA indica la dirección del servidor OCSP. Cuando un cliente quiere verificar su estado envía el numero de serie, el picadillo del nombre del emisor y el picadillo del nombre del sujeto al servidor de OCSP. El servidor constata el estado del certificado y responde con el estado actual del certificado. Con esto se elimina la necesidad de descargar toda la lista de certificados revocados y también se permite la revocación casi instantánea de certificados. El problema que surge es que siempre se requiere conectividad de red con el servidor OCSP.

4.5.6. Formato del certificado

En el estándar de X.509 se especifican un conjunto de campos los cuales deben de estar presentes en un certificado para cumplir con el formato, así como un conjunto de campos opcionales los cuales pueden ser usados para proporcionar información adicional. Todos esto campos así como la firma se especifican en ASN.1.

Versiones 1 y 2

La primera versión del estándar X.509 fue publicada en 1988 como parte del estándar X.500. X.509 era empleado como control de acceso a un directorio X.500 y definía el formato del certificado para ese uso. En esta versión, conocida como X.509 v1, se empleaba un formato estático que contenía un nombre de emisor X.400, un nombre de sujeto X.400, el periodo de vigencia, la llave pública del sujeto y la firma de la CA.

Para la versión 2 del formato se incluyeron dos campos: los identificadores únicos del emisor y del sujeto. Pero no se mejoró de ninguna otra forma la versión 1, pronto se vio la necesidad de hacer más flexible el formato para poder incorporar una variedad más amplia de información. Por todo esto se empezó a trabajar en una versión nueva.

Versión 3

La especificación del formato fue revisada en 1996 para incorporar un campo para datos adicionales el cual permite incluir un conjunto más amplio de información en el certificado. Aunque esto parezca una modificación menor da la posibilidad de incluir información útil para la implementación de una PKI, así como la incorporación de identidades que no sean de tipo X.400, como se realiza en [Martínez Silva, 2005] y [Martínez-Silva et al., 2007]. Todos estos campos opcionales permiten la inclusión de políticas de uso de llaves, información de las políticas de la CA, direcciones de servidores para verificación de estado y más información relevante en el certificado.

El formato v3 de X.509 es el más empleado actualmente y es la base para el RFC 3280 de la IETF.

A continuación se presenta la estructura de un certificado X.509 v3 (tabla 4.3)

Versión	La versión del formato empleado.
Número de serie	Número único asignado por el emisor al certificado.
Algoritmo de firma	El algoritmo que se empleó para firmar el certificado.
Emisor	Nombre de la autoridad que emitió el certificado.
Vigencia	Periodo por el cual es válido el certificado.
Sujeto	La identidad que se certifica.
Llave pública del sujeto	La llave pública que se asocia con el sujeto.
Identificador único del emisor	Obsoleto.
Identificador único del sujeto	Obsoleto.
Extensiones	Lista de los atributos adicionales del certificado.
Firma	Los datos del certificado firmados por el emisor.

Tabla 4.3: Estructura de un certificado X.509 v3

CAPÍTULO 5

DNS

What's in a name? That which we call a rose by any other name would smell as sweet.
—WILLIAM SHAKESPEARE, Romeo and Juliet

El Internet (o cualquier red IP) funciona al emplear, ya sea de forma local o global, direcciones IP en cada equipo participante. Pero, sin la habilidad de asignar un nombre a cada recurso se requeriría conocer la dirección IP, por ejemplo, 148.247.102.15 en vez de `www.cs.cinvestav.mx`. Al existir cientos de millones de direcciones IP sería imposible recordar todos los sitios que visitamos.

Para resolver este problema, a mediados de la década de 1970 se creó el concepto de «servidor de nombres» que hacía posible el obtener ciertos atributos de un «recurso nombrado», en este caso la dirección IP de `www.cs.cinvestav.mx`.

La especificación actual del Sistema de Nombres de Dominio (DNS) data de 1987 y se encuentra en los RFC 1034 [Mockapetris, 1987a] y 1035 [Mockapetris, 1987b].

5.1. Dominios y Delegación

El Sistema de Nombres de Dominio emplea una estructura de árbol. Se tiene el nodo raíz —que contiene un punto (.)— el cual tiene como hijos a los dominios de nivel superior (TLD), éstos a su vez tienen como

hijos a los dominios de segundo nivel (SLD) y así sucesivamente tantos niveles como se quiera, cada uno separado por un punto (.).

La raíz

En el uso cotidiano, el nodo raíz (.) normalmente se omite por conveniencia. Pero, existen ocasiones en las que es muy importante incluirlo, normalmente cuando se configura un servidor DNS.

Los TLDs se dividen en los siguientes tipos:

- Legacy gTLD: dominios de nivel superior genéricos heredados, están disponibles para cualquiera que desee registrarse; por ejemplo: .com, .org, .net
- New gTLD: dominios de nivel superior genéricos nuevos, comenzaron a ser ofrecidos en 2014, se realizó una subasta previa para determinar cuales serían considerados; por ejemplo: .money, .viajes, .vodka.
- ccTLD: dominios de nivel superior de códigos de países, se emplean y están reservados para países y territorios; por ejemplo: .mx, .de, .uk.
- ccTLD IDN: dominios de nivel superior de códigos de países internacionalizados, mismo propósito que los ccTLD pero emplean caracteres no latinos como árabe, chino y cirílico.
- sTLD: dominios de nivel superior patrocinados, parecidos a los gTLD pero el público en general no puede registrarse, solo dependencias u organismos del mismo gremio pueden. Por ejemplo: .aero, .gov, .mil.

Lo que se conoce normalmente como un nombre de dominio, como puede ser, `cinvestav.mx` es realmente la combinación de un SLD y un TLD; escrito de izquierda a derecha con el nivel más bajo de la jerarquía del lado izquierdo y el más alto del lado derecho.

`cinvestav` → SLD
`mx` → TLD

5.2. Autoridad

El concepto de *autoridad* y *delegación* surge del modelo jerárquico de DNS y refleja normalmente la estructura de una organización. Cada nodo dentro de la jerarquía del nombre de dominio es asignada a una *autoridad*, que es, una organización o persona encargada de administrar y operar el nodo. Esa organización o persona administra el nodo de forma *autoritativa*. La *autoridad* de un nodo en particular puede *delegar* autoridad para los niveles inferiores en la jerarquía del nombre de dominio.

La autoridad del nodo raíz lo posee la Corporación de Internet para Nombres y Números Asignados (ICANN, por sus siglas en inglés).

Los gTLDs son administrados de forma autoritativa también por ICANN y delegados a varias empresas. Los ccTLDs son delegados por ICANN a su respectivo país.

Al leer un nombre de dominio de izquierda a derecha se puede seguir cómo ha sido delegado. Por ejemplo:

`www.cs.cinvestav.mx.`

`www` → Nombre del equipo
`cs` ➦ Departamento de Computación
`cinvestav` ➦ Cinvestav CGSTIC
`mx` ➦ NIC México
. ➦ ICANN

5.3. Estructura e Implementación de DNS

La implementación de DNS en Internet refleja exactamente la estructura de delegación descrita previamente. En cada nivel de delegación existen servidores de nombres y la responsabilidad de su operación recae en quien posee la autoridad en ese nivel. Los servidores de nombres raíz son uno de los recursos más importantes en Internet. Cada vez que un servidor DNS es consultado acerca de un dominio el cual no posee información, lo primero que hace es consultar los servidores de nombres raíz. Existen 13 servidores DNS raíz a nivel mundial. Cuando un servidor DNS no puede contestar, *resolver* en la jerga de DNS, la consulta de un nombre, por ejemplo `www.cinvestav.mx`, la consulta se realiza a un servidor raíz, éste lo remite al servidor TLD apropiado, el cual a su vez lo remite al servidor DNS que regresa la respuesta real (autoritativa) (ver figura 5.1).

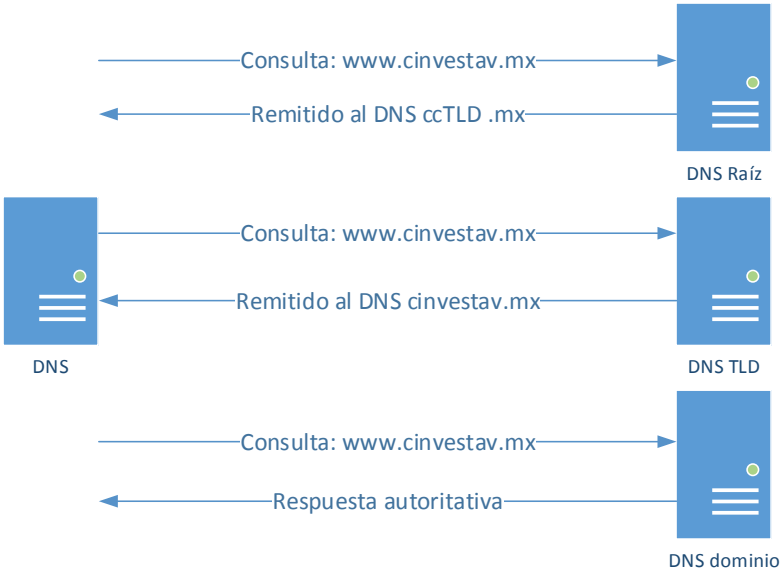


Figura 5.1: Resolución de un nombre

5.4. Rendimiento

Para acelerar los tiempos de respuesta, DNS hace uso de varios niveles de *cache*. Esto significa que las respuestas son almacenadas de forma temporal. Al llegar una solicitud de resolución para el mismo sitio el *cache* se revisa primero, si la información está disponible se regresa de forma directa. Con esto se evita el

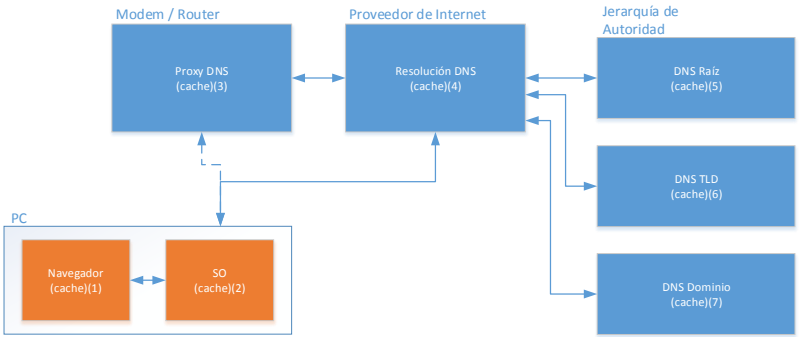


Figura 5.2: Niveles de cache en DNS

realizar comunicaciones innecesarias y se provee una respuesta más rápida. En la figura 5.2 se muestran los diferentes niveles de *cache*.

5.4.1. Envenenamiento de *cache*

El envenenamiento de *cache* de DNS es un ataque en el cuál un usuario solicita la resolución de un nombre a su servidor de DNS, este servidor a su vez solicita la resolución a un servidor autoritativo, pero, un atacante provee una respuesta falsa. De esta manera el usuario se conectaría a una dirección que el atacante maneja, el cuál podría modificar la información que se despliega, espiar y redirigir a al usuario a otro sitio.

El servidor de DNS del usuario a su vez guardaría la respuesta por el tiempo especificado dentro del *cache*, de tal suerte que cualquier otro usuario que solicite la misma resolución obtendrá la dirección IP incorrecta. El *cache* del servidor DNS ha sido envenenado.

5.5. Zonas

Los servidores DNS emplean *zonas*. Una zona es descrita a través de un *archivo de zona*, el cual contiene la traducción de nombres de dominio a entidades operacionales, como equipos, servidores de correo y servicios. Por lo tanto, un archivo de zona describe, empleando registros de recursos (RR), la sección del dominio manejada por un servidor DNS. El formato de los archivos de zona y los RR se encuentra estandarizado en el RFC 1035 [Mockapetris, 1987b]. En la figura 5.3 se muestra un ejemplo de un archivo de zona.

En general un archivo de zona contiene las siguientes directivas y RRs [Aitchison, 2011]:

- Directiva `$TTL` : Establece el tiempo de vida para la zona, el cual es el tiempo que otro servidor DNS lo mantendrá en cache.
- Directiva `$ORIGIN` : El nombre de dominio de la zona.
- RR *Start of Authority (SOA)*: Define las características globales de la zona.
- RR *Name Server (NS)*: Especifica el nombre de los servidores autoritativos para la zona.
- RR *Mail Exchanger (MX)*: Especifica los servidores de correo de la zona.

- RR *Address (A)*: Empleado para definir las direcciones IPv4 de los equipos que existen en la zona y que requieren ser publicados.
- RR *Canonical Name (CNAME)*: Define un alias para otro RR ya existente.
- RR *TXT*: Texto arbitrario asociado a la zona.

```
$ORIGIN example.com
$TTL 86400
@     IN     SOA    dns1.example.com.     hostmaster.example.com. (
2001062501 ; serial
21600      ; refresh after 6 hours
3600       ; retry after 1 hour
604800     ; expire after 1 week
86400 )    ; minimum TTL of 1 day

IN     NS     dns1.example.com.
IN     NS     dns2.example.com.

IN     MX     10     mail.example.com.

IN     A       10.0.1.5

server1      IN     A       10.0.1.5
server2      IN     A       10.0.1.7
dns1         IN     A       10.0.1.2
dns2         IN     A       10.0.1.3

ftp          IN     CNAME   server1
mail         IN     CNAME   server1
www          IN     CNAME   server2

_domkey      IN     TXT     "1dice8EMZmqKvrGE4Qc9bUFf9PX3xaYDp"
```

Figura 5.3: Un archivo de zona

CAPÍTULO 6

BITCOIN

All of the books in the world contain no more information than is broadcast as video in a single large American city in a single year. Not all bits have equal value.

—CARL SAGAN

Bitcoin es una colección de conceptos y tecnologías que forman la base de un sistema de dinero electrónico. Las unidades de de esta criptomoneda se denominan bitcoins y se emplean para almacenar y transmitir valor entre participantes de la red. Los usuarios de Bitcoin se comunican empleando el protocolo de Bitcoin principalmente a través de Internet. La implementación de Bitcoin es de código abierto, por lo que cualquiera puede emplearla, estudiar el código fuente y realizar su propio sistema derivado.

6.1. El dinero

La palabra *dinero*, como se emplea de forma coloquial, tiene muchos significados, normalmente se le confunde con ingreso o patrimonio. Por ejemplo, alguien puede decir: «Los neurocirujanos ganan mucho dinero», significando esto que tienen un ingreso anual alto; o puede decir: «Bill Gates tiene mucho dinero»,

por que su patrimonio —el valor total de sus bienes— es de 78 mil millones de dólares[1].

Por otro lado, para los economistas el dinero tiene una definición muy específica:

> Cualquier cosa generalmente aceptada en el pago de bienes, servicios y deudas.

6.1.1. Funciones del dinero

No importando cuál sea su materialización, el dinero cumple tres funciones principales en la economía [Mishkin, 2004]:

Medio de cambio

En casi todas las transacciones de mercado de nuestra economía, el dinero como circulante[2] o cheques es un *medio de cambio*; se usa para pagar bienes, servicios y cancelación de deuda. Su uso como medio de cambio promueve la eficiencia económica, porque minimiza el tiempo utilizado para intercambiar bienes y servicios. Para saber por qué, imaginemos una economía basada en el trueque, donde los bienes y servicios se intercambian en forma directa por otros.

Daniel, profesor de programación, sólo sabe hacer bien una cosa: dar magníficas clases de programación. En una economía basada en el trueque, si quiere alimentarse, debería encontrar a un agricultor que no sólo produzca los alimentos, sino que también esté interesado en aprender programación. Como es de suponerse, dicha búsqueda sería difícil y tomaría mucho tiempo, y Daniel pasaría más tiempo buscando a un campesino ávido de aprender programación que el que le dedicaría a la enseñanza; incluso tal vez tendría que dedicarse él mismo a la agricultura. Aún así, podría morir de hambre.

El tiempo que se utiliza al intercambiar bienes o servicios se denomina *costo de transacción*. En una economía basada en el trueque, los costos de transacción son altos, porque las personas tienen que satisfacer una «doble coincidencia de necesidades»: tienen que encontrar a alguien que tenga el bien o servicio que desean, y que necesite el bien o servicio que ofrecen.

Si introducimos el dinero en el mundo de Daniel entonces él puede dar clases a cualquiera que esté dispuesto a pagar por asistir a sus clases, y acudir luego con cualquier granjero a comprar los alimentos que necesita con el dinero que se le haya pagado por sus servicios. El problema de la doble coincidencia de necesidades se elimina y Daniel ahorra una gran cantidad de tiempo, que ahora utiliza para hacer lo que mejor sabe hacer.

Como se puede ver, el dinero promueve la eficiencia económica al eliminar gran parte del tiempo que se utiliza en el intercambio de bienes y servicios. También promueve otro tipo de eficiencia al permitir que las personas se especialicen en lo que mejor saben hacer. Es, por lo tanto, esencial en la economía, permite operar de una manera más fluida mediante la disminución de costos de transacción, y fomenta la especialización y la distribución del trabajo.

La necesidad del dinero es tan grande que casi todas las sociedades, excepto las más primitivas, lo inventaron. Para que algo funcione eficazmente como dinero debe de cumplir ciertos requisitos:

- ✓ Debe ser fácil de estandarizar, volviendo simple el determinar su valor.
- ✓ Debe ser aceptado ampliamente.
- ✓ Debe ser divisible, para que sea fácil «dar cambio».

[1] Según Forbes.com LLC; consultado el 18 de marzo de 2015 en `http://www.forbes.com/profile/bill-gates/`
[2] Monedas y billetes

✓ Debe ser fácil de portar.

✓ No debe deteriorarse fácilmente.

Las formas de dinero que satisfacen estos criterios han tomado formas poco usuales a lo largo de la historia de la humanidad, como los collares que usaban los indios americanos, el tabaco y el whisky usados por los primeros colonos norteamericanos, e incluso cigarros, usados por los prisioneros de la Segunda Guerra Mundial. La diversidad de las formas del dinero que se han desarrollado a lo largo de los años es un legado de la inventiva de la raza humana, como lo es el desarrollo de herramientas y de los idiomas.

Unidad de cuenta

El segundo papel del dinero es proporcionar una *unidad de cuenta*; es decir, se usa para medir el valor de la economía. Medimos el valor de los bienes y servicios en términos de dinero. Una vez más tomaremos como referencia una economía de trueque. Si la economía tiene solo tres bienes, entonces necesitamos conocer tres precios que nos indiquen cómo intercambiar un bien por el otro. Para cualesquiera N bienes tendríamos que conocer $N(N-1)/2$ precios, lo cual produciría un costo de transacción muy alto. Este problema es fácilmente solucionado con el empleo de dinero en la economía y haciendo que todos los precios se coticen en términos de las unidades de ese dinero.

Podemos ver que el uso del dinero como unidad de cuenta reduce los costos de transacción de una economía, mediante la disminución del número de precios que necesitan considerarse. Los beneficios de esta función del dinero crecen a medida que la economía es más compleja.

Depósito de valor

El dinero también funciona como un (medio de) *depósito de valor*; es un depósito del poder adquisitivo a través del tiempo. El depósito de valor se emplea para guardar el poder adquisitivo, desde el momento en que se recibe el ingreso hasta el momento en que se gasta. Esta función es útil, porque la mayoría de nosotros no queremos gastar nuestro ingreso inmediatamente después de recibirlo, sino que preferimos esperar hasta que tengamos el tiempo o el deseo de ir de compras.

El hecho de qué tan bueno sea el dinero como medio de depósito de valor depende del nivel de precios, por que su valor es fijo en términos del nivel de precios. Por ejemplo, una duplicación de todos los precios significa que el valor del dinero cae a la mitad; recíprocamente una reducción del 50 % de todos los precios significa que se duplica el valor del dinero. Durante las etapas inflacionarias, cuando el nivel de precios está aumentando con rapidez, el dinero pierde valor rápidamente, y las personas se rehúsan a almacenar su riqueza usando este medio.

6.1.2. Sistemas de pago

Se puede tener un mejor entendimiento de las funciones del dinero y sus materializaciones a través del tiempo al analizar la evolución de los sistemas de pago y la manera en que se realizan las transacciones dentro de la economía. Los sistemas de pago han evolucionado durante siglos, y con ellos la forma del dinero.

Dinero mercancía

En sociedades primitivas y poco organizadas los bienes que hacían la función de dinero generalmente tenían valor en sí mismos y constituían lo que se ha denominado *dinero mercancía*.

Cuando en una sociedad se emplea dinero mercancía, este se utiliza como medio de cambio y también se compra y se vende como un bien ordinario.

Los metales preciosos, oro y plata esencialmente, han sido con frecuencia mercancías elegidas para hacer las veces de dinero. Dado que tienen un elevado valor en usos no monetarios, se puede tener un alto poder de compra sin llevar mucho peso. Por otro lado, las piezas de oro y plata son duraderas y fácilmente almacenables. Asimismo, se pueden dividir sin mucha dificultad y la calidad de las mismas es relativamente fácil de identificar. Pero los metales preciosos, como tales presentan la dificultad de que su calidad y pureza, así como su peso, deben ser evaluados en cada intercambio. Con la acuñación de monedas se minimizan estos inconvenientes, al estampar la autoridad competente su sello como garantía del peso y de la calidad de la moneda [Mochón, 1995].

Dinero fiduciario

El dinero mercancía y, en particular, el dinero metálico, debido a las dificultades presentadas, fue sustituido por el *dinero papel de pleno contenido*, esto es, certificados de papel que estaban respaldados por depósitos de oro o plata de igual valor al de los certificados emitidos. Posteriormente se ha llegado a un sistema como el actual, en el cual el dinero papel no tiene ningún respaldo en términos de metales preciosos, y lo mismo ocurre con el dinero en forma de monedas. Este ha evolucionado hacia el *dinero fiduciario* o *dinero de curso legal*, esto es, papel moneda decretado por los gobiernos como moneda de curso legal (lo cual significa que debe aceptarse legalmente como pago de deudas), pero que no es convertible en metales preciosos. El papel moneda tiene la ventaja de ser más ligero que las monedas o que los metales preciosos, aunque sólo puede aceptarse como medio de cambio si hay suficiente confianza en las autoridades que lo emiten, y si su impresión ha alcanzado una fase suficientemente avanzada que haga sumamente difícil su falsificación.

Cheques

La principal desventaja de los billetes y las monedas es que se pueden robar con facilidad y que resulta caro transportarlos en grandes cantidades debido a su volumen. Para combatir este problema, con el desarrollo de la banca moderna sobrevino otra etapa en la evolución de los sistema de pago: la invención de los *cheques*.

Un cheque es una instrucción que usted le da a su banco para transferir el dinero de su cuenta a la de otra persona, cuando dicha persona deposita el cheque. Los cheques permiten que las transacciones ocurran sin tener que cargar grandes cantidades de dinero. La invención del cheque fue una gran innovación que mejoró la eficiencia de los sistemas de pagos. Con frecuencia, los pagos hechos de un lado a otro se cancelan; sin los cheques, esto implicaría el desplazamiento de mucho dinero. Con ellos, los pagos se anulan mediante la cancelación de los cheques y no es necesario desplazar el dinero. El uso de cheques reduce los costos de transporte asociados con el sistema de pagos y mejora la eficiencia económica. Otra ventaja de su uso es que pueden girarse para cualquier cantidad que no exceda el saldo de la cuenta, haciendo más fácil las transacciones por sumas grandes.

Sin embargo existen dos grandes problemas con ellos. En primer lugar, se requiere de cierta cantidad de tiempo para trasladar los cheques de un lugar a otro, lo cual es particularmente problemático si se está tratando de pagarle a alguien que se encuentra en otro lugar y se quiere hacerlo con rapidez. Segundo, el papeleo para procesarlos es costoso.

Pagos electrónicos

Supongamos que usted pasa su tarjeta de crédito o débito en el supermercado. La máquina que lee la tarjeta primero verifica con el banco emisor de la tarjeta que usted cuente con suficiente saldo o línea de crédito. Si la compra es autorizada, entonces, su banco deduce el monto de la compra de su cuenta y envía un mensaje al procesador de pagos para que se abone la cantidad correspondiente a la cuenta del vendedor.

Como se puede ver los *pagos electrónicos* también disparan una serie de transacciones entre bancos. Estas ocurren a través de redes de computadora que conectan a las diferentes instituciones [Ball, 2011].

Las tarjetas de crédito/débito están diseñadas para realizar pagos en un ambiente de venta al detalle. Esto significa que los pagos solo pueden ser realizados de un tarjetahabiente a un negocio el cual se ha registrado previamente para aceptar pagos con tarjetas. Todos los costos asociados a las transacciones con tarjetas son absorbidos por los negocios. El tarjetahabiente sólo ve el monto de la transacción en su estado de cuenta, pero los negocios pagan un pequeño porcentaje del monto de la transacción el cual se divide entre el banco y el procesador de pagos.

Aparte del uso de tarjetas, los pagos electrónicos han evolucionado en gran medida en los últimos años debido a la gran penetración del Intenet y el uso de los teléfonos inteligentes. Los bancos actualmente ofrecen a sus clientes el poder realizar transacciones desde sus hogares y oficinas a través de sus portales de banca en línea. También es posible realizar movimientos desde las aplicaciones móviles de los bancos, así como realizar pagos empleando tecnología NFC[3] [Ondrus and Pigneur, 2009].

Dinero electrónico

El Banco Central Europeo define el dinero electrónico de la siguiente manera [ECB, 1998]:

> Almacén electrónico de valor monetario en un dispositivo técnico que puede ser usado ampliamente para realizar pagos a terceros diferentes del emisor sin la necesidad de involucrar cuentas bancarias en la transacción, pero actuando como un instrumento prepagado para quien lo posee.

Dependiendo del país alrededor del 75 % y el 95 % de todas las transacciones son realizadas en efectivo, aunque la mayoría son por montos bajos. Es difícil señalar de forma precisa las propiedades del dinero en efectivo las cuales lo vuelven atractivo, pero indudablemente algunas de ellas son [O'Mahony et al., 2001]:

- **Recepción**: el efectivo es casi universalmente aceptado como una forma de pago sin importar el monto de la transacción.
- **Pago garantizado**: al entregarse el efectivo físicamente se completa la transacción y no hay riesgo de que el pago sea anulado posteriormente.
- **Sin cargos**: el efectivo puede ser entregado de una persona a otra sin cargos. No es necesaria una autorización y, por lo tanto, no se requiere de comunicación alguna.
- **Anonimato**: muchas otras formas de pago dejan un registro de una transacción, enlazando a los participantes. El efectivo permite realizar transacciones anónimas. Aparte de ser esto una ventaja para los criminales, es útil para un consumidor honesto el cual se preocupa de la habilidad de las grandes organizaciones de seguir sus movimientos y conocer su estilo de vida.

En 1982, David Chaum [Chaum, 1983] se da cuenta que las transacciones electrónicas que usan tarjetas de débito/crédito implican un problema con respecto al anonimato de los compradores. Chaum propone una

[3] Near Field Communication

forma de realizar pagos electrónicos de manera anónima e introduce el concepto de dinero electrónico, en el cual garantiza el anonimato del comprador.

Propiedades de un sistema de dinero electrónico

Para que un sistema de dinero electrónico pueda considerarse como tal, debe cumplir con ciertas propiedades. Todas ellas hacen que el sistema sea de menor o mayor calidad dependiendo de cuantas de estas propiedades cumpla el sistema. A continuación se describen las propiedades que serían las ideales del dinero electrónico tal y como fueron propuestas en [Okamoto and Ohta, 1992]:

- ✓ **Independencia**: la seguridad del dinero electrónico no puede depender de ninguna condición física. El dinero debe ser enviado a través de la red, por lo que su seguridad no puede depender de que dicha red sea segura.
- ✓ **Seguridad**: el dinero no puede ser copiado ni reutilizado. Debido a que estamos hablando de una moneda electrónica, ésta estaría constituida de bytes los cuales digitalmente podrían ser copiados y reutilizados sin ningún problema. Por lo que se deben de establecer mecanismos con los que se pueda determinar la autenticidad y la reutilización de dichas monedas.
- ✓ **Privacidad**: se debe garantizar el anonimato del comprador, siempre y cuando las transacciones sean válidas. Cuando un comprador use monedas electrónicas no debe ser posible conocer su identidad a través de sus compras. Pero si éste intentara realizar algún tipo de fraude dándole un mal uso a sus monedas electrónicas, el banco será capaz de obtener su identidad para después realizar las actividades legales en contra de éste.
- ✓ **Pago fuera de línea**: las transacciones deben ser realizadas fuera de línea. Cuando una transacción se realice entre comprador y vendedor, el vendedor no debería de estar conectado con el banco para verificar el pago del comprador.
- ✓ **Transferibilidad**: el dinero puede ser transferido a otros. Esta propiedad permitirá a un usuario transferir sus monedas a otros quienes más adelante podrán usar dichas monedas sin ningún problema.
- ✓ **Divisibilidad**: una «pieza» de dinero puede ser dividida en otras de menor denominación. Esto permite que los pagos no requieran un número exacto de monedas electrónicas y así disminuir tanto el tráfico como la cantidad de operaciones que se deben de hacer para validar cada una de las monedas.

Ecash

Una de las primeras compañías en lanzar un esquema de dinero electrónico fue DigiCash, la cual tenía cedes en Estados Unidos y Holanda. La compañia fue fundada por David Chaum quién fue uno de los pioneros en el campo del dinero electrónico y ha sido considerado por algunos como «el padre del dinero electrónico».

Ecash fue desarrollado por DigiCash para hacer posible un esquema de dinero electrónico totalmente anónimo y seguro que pudiera ser usado en Internet. Se dice que Ecash es totalmente anónimo debido a que los clientes reciben las monedas del banco de tal forma que el banco no conoce el número de serie de la moneda que entregó. Las monedas pueden ser gastadas de forma anónima con un vendedor, y aunque el vendedor y el banco estén coludidos no es posible identificar al cliente. Para proveer seguridad al sistema se emplea de forma extensiva criptografía tanto simétrica como asimétrica.

En 1995 el banco Mark Twain de San Luis, Missouri, en los Estados Unidos, comenzó a emitir monedas de Ecash las cuales estaban valuadas a la par del dólar, es decir 0.10 Ecash valía $0.10 USD. Esto se mantuvo por tres años hasta que en noviembre de 1998 DigiCash se declaró en bancarrota.

Prevención de doble gasto

Al ser las monedas de Ecash datos que pueden ser copiados, es necesario prevenir que las monedas puedan ser usadas más de una vez. Ya que el banco no puede ver el número de serie de la moneda que entrega, no puede guardar una relación al emitirlas. Aunque se provee total confidencialidad, la labor del banco para prevenir el doble gasto se dificulta.

Para asegurar que una moneda sea usada sólo una vez el banco emisor debe de registrar cada moneda que es depositada de regreso al banco. Lo cuál genera rápidamente una extensa lista de números de serie usados. Una moneda válida debe:

- Estar firmada por el banco.
- Ser usada dentro de su periodo de vigencia.
- No estar en la base de monedas usadas.

El tercer requerimiento únicamente puede ser validado por el banco emisor que mantiene la base de datos, por lo tanto, las monedas deben de ser enviadas a éste para su verificación en línea cada que se realiza una transacción.

Confidencialidad

Con Ecash sólo el cliente (quien paga) es anónimo. El vendedor (receptor) debe acudir con el banco para validar las monedas. Si el cliente está dispuesto a cooperar también es posible probar que un cliente realizó un pago.

NetCash

NetCash [Medvinsky and Neuman, 1993] [Neuman and Medvinsky, 1995a] es un sistema de dinero electrónico en línea desarrollado en la universidad del Sur de California. Está formada por un cúmulo de servidores de divisa los cuales crean y emiten monedas a los usuarios del sistema. Los usuarios adquieren las monedas a través de cheque electrónico. El sistema se denomina en línea debido a que al recibir una moneda es necesario verificar con el emisor que se encuentre vigente y no haya sido gastada. Cada moneda posee un número de serie identificable y sólo se proporciona un nivel limitado de anonimato a través de un mecanismo en el cual los clientes intercambian sus monedas de forma anónima con los servidores por otras nuevas.

NetCash es un sistema en el cual los usuarios pueden recibir y enviar pagos. Está basado totalmente en software y emplea criptografía tanto simétrica como asimétrica.

Modelo

El ambiente de NetCash consiste de compradores, vendedores y servidores de divisa (ver figura 6.1). Ya que existen varios servidores de divisa, un usuario puede elegir uno cercano a su locación y en el cual confíe. Un servidor de divisa provee los siguientes servicios a sus usuarios:

- Verificar monedas, para prevenir doble gasto.
- Emitir monedas a cambio de un pago a través de cheque electrónico.
- Compra de las monedas, entregando un cheque electrónico.
- Intercambiar monedas válidas por nuevas, para proveer un grado de anonimato.

Para convertir dinero fiduciario a dinero electrónico y viceversa, NetCash emplea el sistema de pago NetCheque [Neuman and Medvinsky, 1995b], también desarrollado por la universidad del Sur de California y que está basado en Kerberos [Steiner et al., 1988], el cual permite emitir cheques electrónicos entre cuentas de un mismo o diferentes bancos.

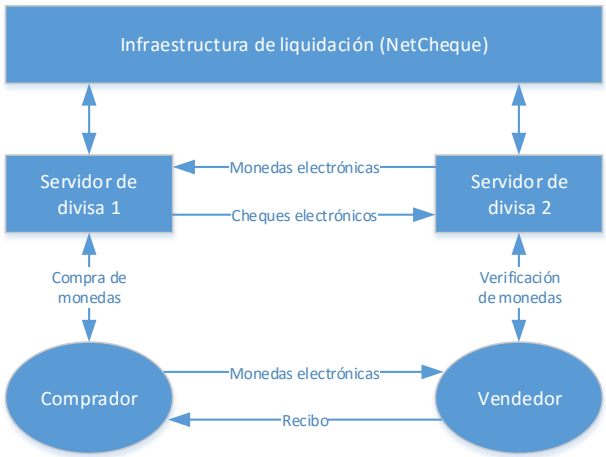


Figura 6.1: El sistema de NetCash

6.2. Bitcoin

Todos los modelos de dinero electrónico propuestos hasta antes de 2008 son centralizados, es decir, existe un ente el cual expide el dinero, verifica las transacciones y lleva el registro de los pagos.

En 2008, Satoshi Nakamoto[4] publica «Bitcoin: A Peer-to-Peer Electronic Cash System» [Nakamoto, 2008], donde combina varios conceptos desarrollados por propuestas previas de sistemas de dinero electrónico para lograr un sistema completamente descentralizado. La innovación clave de Bitcoin está en el uso de un sistema distribuido que emplea un algoritmo de «prueba de trabajo», en el que aproximadamente cada 10 minutos se lleva a cabo una «elección», permitiendo a la red descentralizada llegar a un consenso global acerca del estado de las transacciones.

[4]Se desconoce si el nombre de Satoshi Nakamoto es real o un seudónimo, ni si representa a una persona o a un grupo.

6.2.1. El *blockchain*

El *blockchain* es como se le conoce a la bitácora global de Bitcoin. En él se encuentran almacenadas de forma ordenada y con marcas de tiempo todas las transacciones realizadas desde el inicio de Bitcoin. Su motivo de ser es el prevenir que algún individuo trate de gastar más de una vez una moneda electrónica, así como evitar que alguien trate de modificar transacciones ya realizadas.

Cada cliente completo de la red de Bitcoin almacena de forma independiente una copia del blockchain, la cual es verificada de forma independiente. Cuando varios nodos tienen los mismos bloques en su blockchain se dice que están en consenso.

Generalidades

La estructura del blockchain es una lista ligada y ordenada de bloques de transacciones (ver figura 6.2). El picadillo de cada transacción es obtenido, estos son ordenados en pares y se calcula el picadillo de estos de nuevo hasta que quede un único picadillo, el cual se conoce como la raíz Merkle de un árbol Merkle.

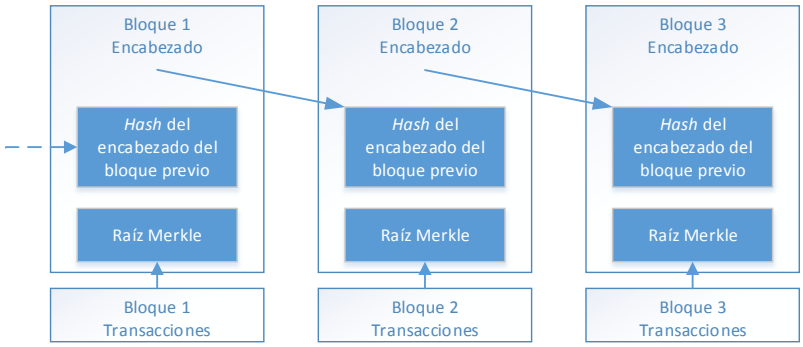


Figura 6.2: Diagrama simplificado del blockchain

La raíz Merkle se almacena en el encabezado del bloque. También cada bloque almacena el picadillo del encabezado del bloque previo, enlazando de esta manera los bloques. Así se previene que un bloque pueda ser alterado sin también tener que modificar todos los bloques posteriores.

También las transacciones son enlazadas una con otra. Los programas que se emplean para realizar operaciones con bitcoins dan la impresión de que los *satoshis*[5] se envían de una cartera a otra, pero, en realidad se mueven de transacción en transacción. Cada transacción gasta los satoshis previamente recibidos en una o más transacciones previas, así la entrada de cada transacción es la salida de una previa (ver figura 6.3).

Cada transacción puede crear múltiples salidas, este sería el caso cuando se hacen envíos a múltiples direcciones, pero la salida de una transacción en específico sólo puede ser usada como entrada una vez. Cualquier intento de usarla de nuevo sería tratar de gastar dos veces los mismos satoshis, algo que no está permitido.

[5]Un satoshi equivale a 0.00000001 bitcoins y es la unidad más pequeña en Bitcoin.

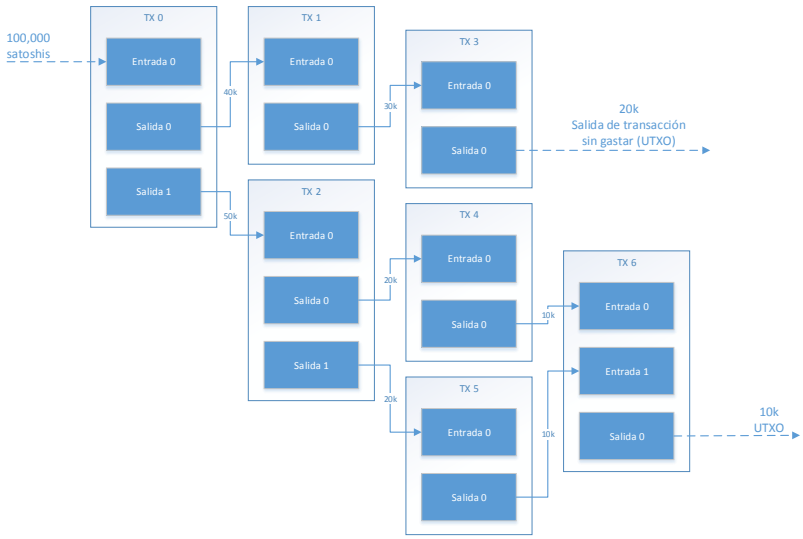


Figura 6.3: Esquema de cómo se realizan las transacciones en Bitcoin

Las salidas están identificadas por un identificador de transacción ,TXID (*transaction identifier* en inglés), los cuales son el picadillo de una transacción firmada.

Ya que cada salida sólo puede ser gastada una vez, se pueden categorizar las salidas de todas las transacciones en el blockchain como: a) salidas de transacción sin gastar (UTXO, por su acrónimo en inglés) y b) salidas de transacción gastadas. Para que un pago sea válido debe de tener sólo UTXOs como entrada.

Si el valor de las transacciones de salida excede el de entrada la transacción es rechazada —existe una excepción, las transacciones de acuñado—, pero, si la entrada excede la salida, entonces la diferencia puede ser reclamada como una comisión por el *minero* que cree el bloque que contenga la transacción. En la figura 6.3 cada transacción gasta 10000 satoshis menos de los que recibió por lo que se está pagando una comisión de 10000 satoshis.

Árbol Merkle

Llamados así debido a que fueron inventados por Ralph Merkle [Merkle, 1988]. Son árboles binarios de picadillos[6], empleados para la verificación eficiente de integridad de datos. En la figura 6.4 se puede ver un ejemplo. Todas las hojas son calculadas de forma directa al obtener el picadillo de los datos. Los nodos superiores son obtenidos al calcular el picadillo de la concatenación de cada nodo inferior. Dado el caso que la cantidad de unidades de datos sea impar, la unidad final se duplica para así obtener una cantidad par. Por ejemplo, si en la figura 6.4 no existiera d_{11}, entonces d_{10} se duplicaría y se colocaría en su lugar.

La principal ventaja de los árboles Merkle es que al ser modificado un dato no es necesario recalcular todos los picadillos. Si en la figura 6.4 modificamos el contenido de d_{00}, entonces se requeriría regenerar n_{00}

[6]En Bitcoin la función picadillo empleada es: SHA-256(SHA-256()).

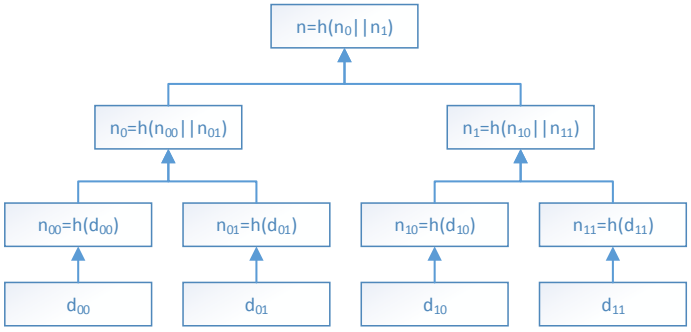


Figura 6.4: Árbol Merkle

así como todos los nodos de esa rama hasta llegar a la raíz. Por lo tanto, el número de picadillos a calcular se incrementa logarítmicamente con el número de bloques de datos.

También se tiene la ventaja de que es fácil verificar que algún dato fue incluido en el árbol. Por ejemplo, si quisiéramos verificar que el dato d_{11} fue incluido en el árbol sólo necesitamos pedir a quién generó la raíz Merkle d_{10} y n_0. Con estos datos podemos por nuestra cuenta generar de nuevo la raíz. En el caso de que coincida, entonces d_{11} fue incluido en el árbol.

Direcciones, llaves y carteras

La posesión de bitcoins se maneja a través de llaves, direcciones y firmas digitales. Las llaves no son almacenadas por la red de Bitcoin, sino por los usuarios en archivos o bases de datos, denominadas carteras. Las llaves son totalmente independientes del protocolo de Bitcoin, por lo que pueden ser generadas de forma independiente y sin necesidad de acceso al blockchain.

Cada transacción en Bitcoin requiere de una firma válida para poder ser incluida en el blockchain, para poder realizarla se requiere de la llave correspondiente. Las llaves se manejan en pares: una pública y una privada (consultar 3.4).

secp256k1

Las llaves públicas y privadas de bitcoin están basadas en criptografía de curva elíptica. Se emplea la curva `secp256k1`, ésta se encuentra definida en el estándar SECG [SECG, 2009] con los parámetros de dominio especificados por la sextúpla $T = (p, a, b, G, n, h)$ sobre $\mathbb{F}_p$.

Donde:
$p =$ `FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F`
$= 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$

La curva $E : y^2 = x^3 + ax + b$ sobre $\mathbb{F}_p$ esta definida por:
$a = 0$
$b = 7$

El punto base G en forma comprimida es:
$G =$ `02 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798`

Sin comprimir:
$G =$ `04 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798 483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8`

Finalmente, el orden n de G y el cofactor son:
$n =$ `FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141`
$h = 1$

Representación de los puntos

Un punto en la curva elíptica consiste de un par de coordenadas (x, y), para representar un punto existen dos maneras:

- **Sin comprimir**: se emplea el prefijo `04` seguido de dos números de 256 bits, la coordenada x y la coordenada y.
- **Comprimido**: debido a que la curva está dada por una ecuación ($y^2 = x^3+7$ en el caso de `secp256k1`) cada punto en la curva representa una solución a esta, de tal manera que si conocemos la coordenada x podemos obtener la coordenada y. Esto permite ahorrar el 50 % de espacio al representar un punto. Ya que al evaluar la coordenada x obtendremos y^2 es necesario indicar el signo de la coordenada y a emplear, esto se logra empleando el prefijo `02` si es positivo y `03` si es negativo.

Llaves

Para obtener un par de llaves primero se debe seleccionar un número k de 256 bits al azar. El número debe de encontrarse en el rango $[1, n - 1]$, donde n es el orden de la curva `SECP256k1`; y k es la llave privada.
Una vez que se tiene la llave privada se obtiene la llave pública K_{pub} de la siguiente manera: se realiza la multiplicación escalar $K_{pub} = k * G$ donde G es el punto base de `SECP256k1`.

Direcciones

Se puede producir una dirección a partir de una llave pública, para esto se emplean las funciones picadillo SHA-256 [NIST, 2012] y RIPEMD-160.

El procedimiento es el siguiente (ver figura 6.5):

1. Se toma la llave pública y se le calcula su picadillo empleando SHA-256. Se obtiene una salida de 256 bits.
2. Al picadillo obtenido se le calcula a su vez su picadillo empleando RIPEMD-160, el resultado es la dirección correspondiente. Se obtiene una salida de 160 bits.

Base58Check

Las direcciones de Bitcoin casi siempre se presentan en la codificación *Base58Check*, esto para facilitar su lectura y evitar errores. Base58Check emplea un conjunto de 58 caracteres alfanuméricos fáciles de

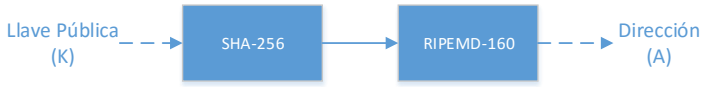


Figura 6.5: Proceso para generar una dirección a partir de una llave pública

distinguir[7] junto con un código de verificación de errores. *Base58Check* no sólo se emplea para las direcciones, se emplea cada que se desea que un usuario pueda leer y transcribir de forma correcta un número [Antonopoulos, 2014].

Carteras

En Bitcoin al referirse a una cartera, se hace referencia ya sea a un archivo o a un programa. Los programas generan llaves públicas para poder recibir satoshis y usan sus llaves privadas correspondientes para poder gastarlos. Los archivos de cartera almacenan las llaves privadas e información relacionada a las transacciones realizadas.

La única funcionalidad esencial de un programa de cartera es el poder recibir y gastar satoshis. Pero no es necesario que la implementación realice las dos cosas, se puede tener un programa que distribuya las direcciones para poder recibir satoshis y otro programa el cuál firme las transacciones que gasten esos satoshis.

Los programas de cartera también tienen la necesidad de interactuar con la red P2P, de la cual obtienen la información del blockchain y a través de la cual envían las nuevas transacciones. Aunque puede ser un programa diferente al que distribuye las direcciones y al que firma las transacciones.

Esto nos deja con tres partes necesarias, pero separables, en un sistema de cartera: un programa de distribución de direcciones, un programa de firma y un programa de red P2P. A continuación se describen las combinaciones más comunes en los diferentes programas de cartera:

Carteras completas

La cartera más simple es la cual realiza todas las operaciones necesarias: genera llaves privadas y públicas, crea las direcciones correspondientes, verifica si se han realizado transacciones hacia las direcciones administradas, crea y firma transacciones desde las direcciones administradas y comunica a la red P2P las transacciones firmadas. La principal ventaja de este tipo de carteras es que son fáciles de usar, un programa hace todo lo que el usuario necesita para recibir y gastar satoshis. Por otro lado su principal desventaja es que almacenan la llave privada en un dispositivo conectado a Internet, lo cual implica el riesgo de compromiso.

Carteras de firma

Para disminuir el riesgo de robo, las llaves privadas pueden ser generadas y almacenadas en un programa de cartera independiente el cuál este instalado en un sistema más seguro. Las carteras de firma se emplean en conjunto con otra cartera que posea conectividad a la red P2P. Ambas carteras pueden ser instaladas

[7]Los caracteres empleados son: 123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz

en un mismo equipo o en equipos diferentes. Dependiendo de los requerimientos se puede llegar hasta la instalación de la cartera de firma en equipos aislados que sólo se encienden para firmar transacciones. La principal desventaja de este tipo de carteras es el trabajo adicional que trae el tener que transferir de un equipo a otro las transacciones a firmar y posteriormente transferir la versión firmada para su propagación en la red P2P.

Carteras en hardware

Las carteras en hardware son dispositivos dedicados los cuales implementan una cartera de firma. Al ser dispositivos diseñados específicamente pare este propósito eliminan muchos de los riesgos presentes en los sistemas operativos de uso general. Un ejemplo de estas son las comercializadas por la empresa checa SatoshiLabs.

Carteras de distribución

Cuando se requiere una cartera en un ambiente difícil de asegurar, como lo es un servidor web, se puede emplear una cartera la cual sólo distribuya direcciones. Las dos formas más comunes de lograr esta funcionalidad mínima son:

- Tener una base de datos con direcciones pregeneradas, cada que se entrega un dirección se marca como usada para evitar el reuso.
- Emplear una llave pública padre para generar llaves públicas hijas, para evitar el reuso cada que se entrega una llave se registra esta o se mantiene un apuntador al índice de la llave entregada.

Bitcoin Core

Bitcoin Core, también conocido como cliente Satoshi, es una cartera completa de Bitcoin y representa la columna vertebral de la red. Bitcoin Core es el código fuente de Bitcoin y es la implementación de referencia de la red Bitcoin. Es el único programa que implementa totalmente el protocolo Bitcoin. Se puede descargar de forma gratuita desde la página `https://bitcoin.org/en/download`.

Los cambios que se implementan en Bitcoin Core son propuestos por la comunidad a través de una *propuesta de mejora a Bitcoin* (BIP, por sus siglas en inglés). Estos pueden ser consultados en la página `https://github.com/bitcoin/bips`.

♪ Can't Get Enough of ... ♪

Si desea instalar Bitcoin Core debe tener en cuenta que al ser un una cartera completa ésta guarda una copia local del blockchain, lo cual requiere de aproximadamente 55 GB (5 de noviembre de 2015) de espacio en disco duro y seguirá aumentando indefinidamente con el tiempo.

Prueba de trabajo

El blockchain es mantenido de forma colaborativa por todos los integrantes de la red, por lo que Bitcoin requiere que se pruebe que en la creación de cada bloque se invirtió una cantidad significativa de trabajo.

Esto para asegurar que usuarios malintencionados que traten de modificar bloques anteriores se vean obligados a realizar una cantidad excesivamente mayor de trabajo que los usuarios honestos que quieren agregar un nuevo bloque. Al depender un bloque de los bloques previos, es imposible modificar un bloque dado sin tener que modificar todos los bloques subsecuentes. Esto da como resultado que el costo de modificar un bloque se incrementa con cada nuevo bloque que es agregado.

El algoritmo de prueba de trabajo empleado en Bitcoin aprovecha la naturaleza aparentemente aleatoria de las funciones picadillo. Para probar que se realizó trabajo al crear un nuevo bloque, se debe crear un picadillo del encabezado del bloque que no exceda cierto valor. Por ejemplo, si el máximo valor posible del picadillo es $2^{256} - 1$, podemos probar que intentamos dos combinaciones al producir un valor menor a 2^{255}.

Los nuevos bloques serán agregados al blockchain sólo si el valor es menor o igual al nivel de dificultad. Este valor es consensuado por la red y actualizado cada 2016 bloques[8]. El valor se establece tratando de lograr que con todos los nodos compitiendo por encontrar un valor menor para cada nuevo bloque, se tome aproximadamente 10 minutos el encontrarlo. De forma ideal, el generar 2016 bloques toma 1209600 segundos (dos semanas). Si se tomó menos de dos semanas en generar los 2016 bloques, entonces el valor de la dificultad es incrementado de forma proporcional (con un máximo de 300 %). En el caso de que se haya tardado más de dos semanas, se decrementa el valor de la dificultad de forma proporcional (con un máximo de 75 %).

6.2.2. Transacciones

Las transacciones son la parte central de Bitcoin, todas las demás partes están construidas para asegurar que las transacciones sean creadas, propagadas en la red P2P, verificadas y agregadas al blockchain. Las transacciones son estructuras de datos las cuales almacenan transferencias de valor entre participantes del sistema. Cada transacción es almacenada como una entrada en el blockchain.

Ciclo de vida

El ciclo de vida de una transacción inicia en el momento que la transacción es construida. Después la transacción es firmada una o varias veces para indicar que esta ha sido autorizada de tal forma que un determinado monto de satoshis sean transferidos. Posteriormente la transacción es propagada a la red P2P, donde cada nodo participante verifica y propaga la transacción hasta que esta llega a prácticamente todos los nodos en la red. Finalmente la transacción es validada por un nodo minero y es incluida en un nuevo bloque de transacciones que es agregado al blockchain.

Una vez que se encuentra en el blockchain y ha sido confirmada por un número suficiente de bloques subsecuentes, la transacción es una parte permanente del blockchain y es aceptada por todos los participantes. Los fondos pueden ser ahora transferidos por en nuevo propietario a través de una nueva transacción.

Estructura de la transacción

Una transacción es en esencia una estructura de datos que almacena las transferencias de valor desde un origen, llamado *entrada*, a un destino, denominado *salida*. Las entradas y salidas de una transacción no están

[8]Debido a un error por uno, sólo se toman los valores de 2015 bloques al calcular la dificultad.

asociadas a cuentas o identidades. En su lugar, son un monto de satoshis los cuales sólo pueden ser gastados por el propietario de la dirección de destino, esto mediante el uso de la llave privada asociada a esta. Una transacción tiene la estructura mostrada en la tabla 6.1.

Tamaño	Campo	Descripción
4 bytes	Versión	Reglas a las cuales se apega la transacción
1-9 bytes	Total de entradas	El número de entradas que se incluyen
Variable	Entradas	Una o más entradas de la transacción
1-9 bytes	Total de salidas	El número de salidas que se incluyen
Variable	Salidas	Una o más salidas de la transacción
4 bytes	Bloqueo	Una fecha en formato UNIX o un número de bloques

Tabla 6.1: La estructura de una transacción

Bloqueo de una transacción

El bloqueo indica la fecha mínima en la cual la transacción puede ser agregada al blockchain. Normalmente se emplea 0 para indicar que debe incluirse lo más pronto posible. Si el valor indicado es diferente de cero y menor 500 millones entonces se interpreta como una altura en bloques, queriendo decir esto que la transacción no debe de incluirse antes de ese número de bloque. Si el valor es mayor a 500 millones entonces se interpreta como una fecha en formato UNIX.

Entradas y salidas

El componente principal de una transacción son las UTXOs (salidas de transacción sin gastar). Las UTXOs son montos indivisibles los cuales sólo pueden ser liberados por un dueño específico, están almacenadas en el blockchain y son reconocidas por toda la red. La red lleva el seguimiento de todas las UTXOs, cuando un usuario recibe un monto en satoshis este se almacena en el blockchain como una UTXO. El concepto del balance total de un usuario no existe como tal, sólo existe un conjunto de UTXOs distribuidas a lo largo del blockchain las cuales pueden ser transferidas por el usuario que posea la llave privada. Los programas de cartera calculan el balance al recorrer el blockchain y agregar cada UTXO que le pertenezca al usuario.

Aunque una UTXO pueda tener un valor arbitrario ésta es indivisible. Si una UTXO es mayor al valor que se desea transferir, ésta se debe de consumir en su totalidad y el cambio correspondiente se debe generar en la transacción. Las UTXOs consumidas por una transacción son llamadas entradas y las UTXOs generadas por la misma transacción son llamadas salidas. Esto se puede apreciar en la figura 6.6 donde hay una entrada de 100000 satoshis, se desea transferir 40000 satoshis (salida) y se obtienen 60000 satoshis (salida) de cambio.

De esta manera los satoshis se mueven de un dueño a otro en una cadena de transacciones que consumen y crean UTXOs. Las transacciones consumen las UTXOs al liberarlas a través de una firma y crean una UTXO al asignarla a la dirección de un nuevo propietario. La excepción a la cadena de salidas y entradas es un tipo especial de transacción llamada de *acuñado*, la cual es la primera transacción de cada bloque. Esta transacción es colocada por el minero «ganador» y crea nuevos satoshis que se entregan como recompensa por el minado.

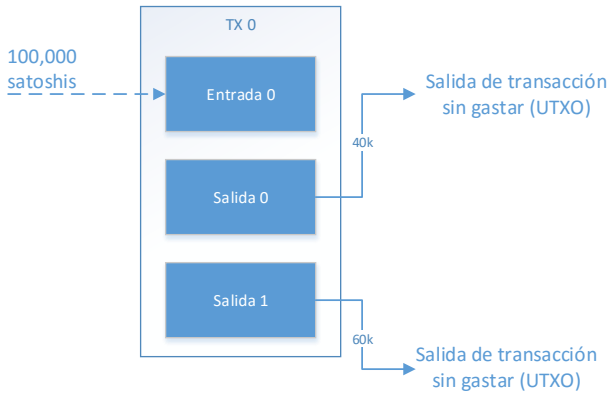


Figura 6.6: Entradas y salidas de una transacción

Salidas

Cada transacción crea salidas, las cuales son almacenadas en el blockchain. Casi todas estas salidas (existe una excepción) crean UTXOs las cuales son reconocidas por toda la red y están disponibles para que el dueño las gaste en un futuro. El transferirle a alguien un monto en satoshis es realmente crear una UTXO asignada a su dirección.

Las salidas de una transacción están formadas por dos partes:

1. Un monto de satoshis.
2. Un *script* de bloqueo, el cual restringe el poder gastar el monto al especificar las condiciones que se deben de cumplir para poder gastar la salida.

A continuación se presenta en la tabla 6.2 la estructura de una salida:

Tamaño	Campo	Descripción
8 bytes	Monto	Cantidad de satoshis
1-9 bytes	Tamaño del script de bloqueo	Tamaño en bytes que tiene el script de bloqueo
Variable	Script de bloqueo	Un script el cual define las condiciones que se deben de cumplir para poder gastar el monto

Tabla 6.2: La estructura de una salida

Las salidas asocian un monto específico de satoshis a un *script* de bloqueo el cual especifica las condiciones que se deben de cumplir para poder gastar ese monto. En la mayoría de los casos el *script* de bloqueo asigna el monto a una dirección, transfiriendo de esta forma los satoshish al nuevo dueño.

Entradas

Descrito de forma simple, las entradas son apuntadores a UTXOs. Estos apuntan a una UTXO específica a través del picadillo de la transacción y el número de secuencia donde la UTXO se encuentra en el blockchain. Para gastar una UTXO la entrada de la transacción también incluye un *script* de desbloqueo que satisface la condición especificada por la UTXO. Este *script* de desbloqueo normalmente consta de una firma la cual prueba la posesión de la dirección que se encuentra especificada en el *script* de bloqueo de la UTXO.

Cuando los usuarios realizan pagos, los programas de cartera seleccionan UTXOs disponibles para construir la transacción. Ya que se tienen las UTXOs los programas de cartera generan los *scripts* de desbloqueo que cumplen las condiciones de cada UTXO. El programa de cartera agrega las referencias a las UTXOs y los *scripts* de desbloqueo como entradas de la transacción. En la tabla se muestra la estructura de una entrada.

Tamaño	Campo	Descripción
32 bytes	Picadillo de la transacción	Apuntador a la transacción que contiene la UTXO a gastar
4 bytes	Índice de la salida	El índice de la UTXO a ser gastada. Inicia en 0.
1-9 bytes	Tamaño del script de desbloqueo	Tamaño en bytes que tiene el script de desbloqueo
Variable	Script de desbloqueo	Un script el cual cumple las condiciones del script de bloqueo de la UTXO
4 bytes	Número de secuencia	No empleado actualmente. Valor 0xFFFFFFFF

Tabla 6.3: La estructura de una entrada

Comisiones

La mayoría de las transacciones incluyen comisiones, las cuales compensan a los mineros por su trabajo de asegurar la red. Las comisiones sirven de incentivo para que los mineros incluyan la transacción en el siguiente bloque, así como desincentivan el abuso de la red al imponer un pequeño costo a cada transacción. Las comisiones son recolectadas por los mineros a la hora que generan un nuevo bloque. Como tal, las comisiones no son obligatorias pero incentivan el procesamiento de la transacción.

Las comisiones son calculadas con base al tamaño en kilobytes de la transacción, el valor de la transacción no es de importancia. El monto de las comisiones es determinado por el mercado. Los mineros asignan prioridades a las transacciones basados en diferentes criterios, entre ellos la comisión, y pueden llegar a incluir transacciones que no otorguen comisión alguna bajo ciertas circunstancias. Aunque normalmente es más probable que una transacción con una buena comisión sea incluida en el siguiente bloque, mientras que una con comisión baja o nula sea demorada y procesada bajo mejor esfuerzo después de varios bloques o no sea procesada.

Como se puede ver en la tabla 6.1 una transacción no posee un campo para especificar la comisión. En vez de esto, las comisiones se encuentran implícitas en la diferencia entre las entradas y las salidas. Cualquier monto que sobre después de que todas las salidas han sido deducidas de las entradas es una comisión.

$$comisión = Suma(entradas) - Suma(salidas)$$

En la figura 6.7 se puede observar una transacción la cual recibe una entrada de 100000 satoshis, realiza un pago de 40000 satoshis y genera 50000 satoshis de cambio, los 10000 satoshis restantes son una comisión al minero que incluya la transacción en un bloque.

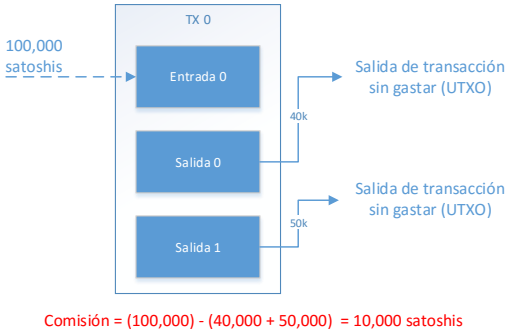


Figura 6.7: Una transacción que incluye comisión

Gardez la monnaie!

Siempre que se construya una transacción es importante incluir una salida con el cambio, en el caso de no incluirla todo el monto que no se encuentre en las demás salidas es considerado una comisión.

Scripts

Los clientes de Bitcoin validan las transacciones al ejecutar un *script* escrito en un lenguaje llamado *Script*. *Script* posee las siguientes características:

- Imperativo.
- No es Turing completo.
- Emplea notación polaca inversa.

Este lenguaje fue hecho específicamente para Bitcoin, es muy simple y limitado, requiere de procesamiento mínimo y puede ser implementado en un amplia gama de dispositivos. El lenguaje no es Turing completo debido a que no incorpora ciclos o controles de flujo complejos, esto hace que no sea posible tener ciclos infinitos o algún tipo de bomba lógica que ocasione un ataque de denegación de servicio en la red de Bitcoin. Todos los scripts carecen de estado, es decir no existe un estado previo o posterior a la ejecución del mismo, por lo tanto toda la información necesaria para ejecutar el script debe estar contenida en él. Si después de

ejecutar el script se tiene un resultado diferente a VERDADERO la entrada es inválida ya que no se ha satisfacido la condición en la UTXO.

El lenguaje ejecuta el script al procesar cada elemento de izquierda a derecha. Los números (constantes) se colocan en la pila. Los operadores meten o sacan uno o más parámetros de la pila, modifican los parámetros y pueden meter un resultado a la pila.

Por ejemplo, la instrucción `OP_ADD` saca dos elementos de la pila, los suma y mete el resultado a la pila.

Los operadores condicionales evalúan una condición y producen un resultado booleano de VERDADERO o FALSO. Por ejemplo, la instrucción `OP_EQUAL` saca dos elementos de la pila y mete un VERDADERO (0x01) si son iguales o un FALSO (0x00) si son diferentes.

En la figura 6.8 se muestra la ejecución del siguiente script (el cual realiza una operación de suma):

```
1 3 OP_ADD 4 OP_EQUAL
```

Aunque la mayoría de los scripts de bloqueo hacen referencia a una dirección, y por lo tanto se requiere probar la propiedad de dicha dirección para poder emplear la UTXO, no es necesario que los scripts sean así de complejos. Cualquier combinación de un script de desbloqueo y uno de bloqueo que finalicen con un valor de VERDADERO en la pila es válido. El ejemplo de la figura 6.8 es totalmente válido para ser incluido en la salida de una transacción.

El script de bloqueo sería:

```
3 OP_ADD 4 OP_EQUAL
```

Este script puede satisfacerse con el script de desbloqueo:

```
1
```

Al realizar la verificación se concatenan los scripts de desbloqueo y bloqueo para tener el script resultante:

```
1 3 OP_ADD 4 OP_EQUAL
```

Transacciones estándar

Después de descubrirse varias vulnerabilidades importantes en las primeras versiones de Bitcoin se incluyó una verificación la cual sólo acepta transacciones de la red si las operaciones de firma y verificación de firma cumplen con un conjunto de plantillas las cuales se consideran seguras y si el resto de las operaciones no se salen de un conjunto de reglas que garantizan un comportamiento adecuado de la red. Esta verificación se llama `IsStandard()` y todas las transacciones que la cumplan se consideran transacciones estándar.

Pago a picadillo de llave pública

La mayoría de las transacciones en Bitcoin son del tipo pago a picadillo de llave pública (P2PKH, por sus siglas en inglés). Estas contienen un script de bloqueo que asigna la salida a una dirección generada a partir de una llave pública. Las direcciónes generadas a partir de una llave pública una vez codificadas con base58check comienzan con 1. La salidas bloqueada por un script P2PKH pueden ser desbloqueados al presentar la llave pública y una firma digital creada por la correspondiente llave privada.

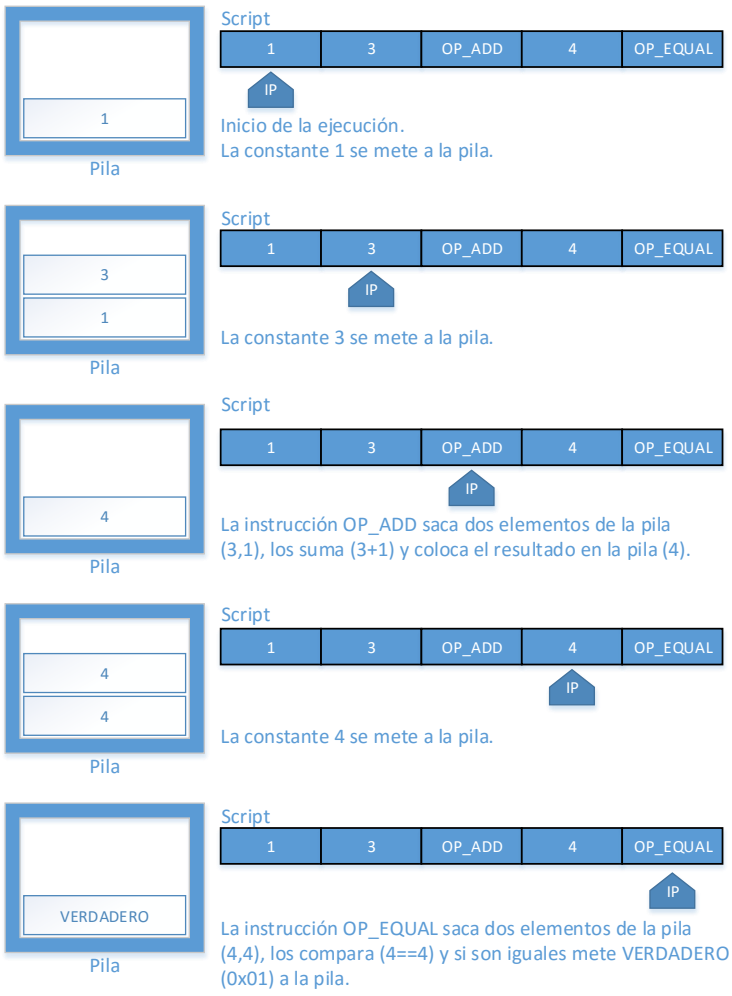


Figura 6.8: Ejecución de un script

En la figura 6.9 se muestra el proceso que se sigue al evaluar los siguientes scripts:
Script de bloqueo:

```
OP_DUP OP_HASH160 <dirección> OP_EQUALVERIFY OP_CHECKSIG
```

Script de desbloqueo:

$$\texttt{<firma}_k\texttt{>}\ \texttt{<K}_{pub}\texttt{>}$$

Pago a llave pública

Es una forma simplificada de los scripts P2PKH, en ella se especifica la llave pública. Era empleada anteriormente, sigue siendo válida pero se recomienda emplear scripts P2PKH para transacciones nuevas.
Script de bloqueo:

$$\texttt{<K}_{pub}\texttt{>}\ \texttt{OP\_CHECKSIG}$$

Script de desbloqueo:

$$\texttt{<firma}_k\texttt{>}$$

Multifirma

Los scripts de multifirma establecen una condición donde N llaves públicas se listan en el script de bloqueo y se deben de proporcionar por lo menos M firmas en el script de desbloqueo. Se tiene el límite de 16 llaves públicas, lo cual significa que se puede elegir cualquier valor $N, M \mid 1 \leq N \leq M \leq 16$. Los valores de M y N se especifican mediante códigos de operación de la forma `OP_1, OP_2, OP_3 ...OP_16` . Debido a un error por uno el script de desbloqueo debe iniciar con `OP_0`. Las firmas deben de proporcionarse en el mismo orden de las correspondientes llaves públicas.
Script de bloqueo:

$$\texttt{<M>}\ \texttt{<K}^A_{pub}\texttt{>}\ [\texttt{K}^B_{pub}]\ [\texttt{K}^C_{pub}]\ \ldots\ \texttt{<N>}\ \texttt{OP\_CHECKMULTISIG}$$

Script de desbloqueo:

$$\texttt{OP\_0}\ \texttt{<firma}^A_k\texttt{>}\ [\texttt{firma}^B_k]\ [\texttt{firma}^C_k]\ \ldots$$

Datos

Al tener Bitcoin usos potenciales más allá de simplemente realizar pagos, varios desarrolladores han empleado el blockchain para otras aplicaciones como notarización y contratos. Por ejemplo, para demostrar la existencia de un documento, su picadillo es usado como dirección en una salida, después de que la transacción es minada esta información estará siempre en el blockchain. De esta manera si alguien pide que se pruebe que un diseño, fórmula, algoritmo, etc. fue creado antes de cierta fecha se puede comprobar obteniendo el picadillo del documento e indicando su posición en el blockchain. Esto produce que se pierdan satoshis ya que no es posible generar un script de desbloqueo, también al ser UTXOs estas son monitoreadas por los nodos de la red lo cual consume recursos. Aparte de hacer que el tamaño del blockchain se incremente con información que no es del sistema. Para proporcionar una alternativa menos destructiva (no se

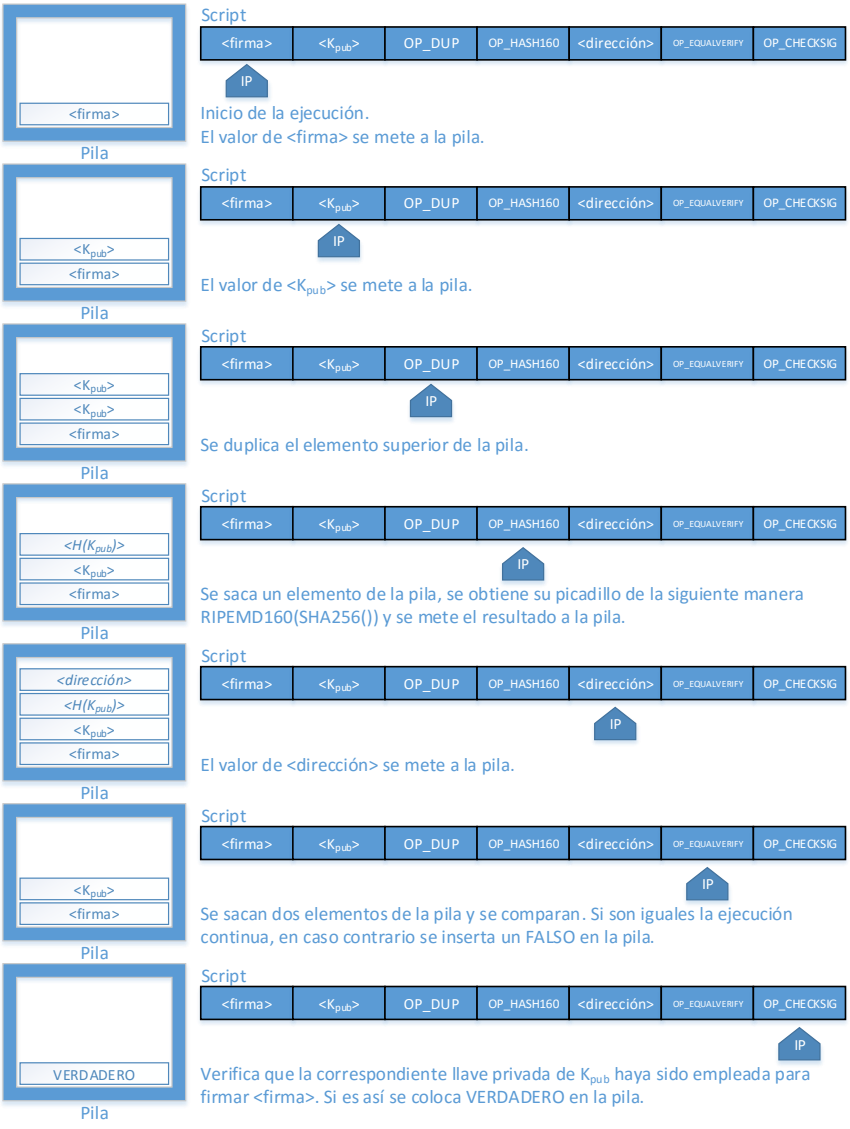


Figura 6.9: Ejecución de un script P2PKH

pierden satoshis) y con la cual se pudiera indicar a los nodos que no requieren procesar la información de esa salida se incluyó en la versión 0.9 de Bitcoin Core la instrucción `OP_RETURN`.

`OP_RETURN` permite incluir hasta 40 bytes de datos. El número de salidas que contienen un script de bloqueo `OP_RETURN` está limitado a una por transacción. Es necesario pagar comisión para que la transacción sea incluida en el blockchain.

Script de bloqueo:

```
OP_RETURN <datos>
```

Piénsalo muy bien

El uso `OP_RETURN` no es recomendado debido a que hace crecer el blockchain con información que no es relevante para Bitcoin, sólo fue creado para evitar el uso de alternativas más destructivas.

Pago a picadillo de script

Este tipo de transacción fue añadida en 2012, el pago a picadillo de script (P2SH, por sus siglas en inglés) permite simplificar el uso de scripts en transacciones complejas. Por ejemplo, aunque el uso de multifirma es muy conveniente, se tienen que proporcionar mucha información a quién realiza el pago: el valor de N, el de M, cada una de las llaves públicas y el orden de las mismas. Todo esto hace que el script de bloqueo se vuelva muy largo y que sea fácil cometer un error; además de que no todas las carteras aceptan el uso de multifirma. Por otro lado, recordemos que las comisiones se basan en el tamaño de la transacción, por lo que se deberá pagar una mayor comisión.

Con P2SH las transacciones complejas se simplifican al sustituir un script por su picadillo. A partir del picadillo se genera una dirección la cual ya codificada en base58check empieza con 3. En el momento que se quiera emplear una UTXO que emplea P2SH se debe presentar el script de bloqueo que genera el picadillo así como el correspondiente script de desbloqueo.

Script de bloqueo:

```
OP_HASH160 <RIPEMD-160(SHA-256(script de bloqueo))> OP_EQUAL
```

Script de desbloqueo:

```
<script de desbloqueo> <script de bloqueo>
```

6.2.3. Estructura de red

La red de Bitcoin se encuentra formada por conexiones punto a punto (P2P) entre cada uno de los nodos, no existe ningún tipo de jerarquía. Cualquier nodo se puede unir a la red con sólo conectarse a un nodo aleatorio de la red.

Aunque en Bitcoin no existen nodos principales, existen nodos los cuales llevan largo tiempo conectados de forma estable, la dirección de estos nodos llamados *nodo!semilla* se incluye en el código de Bitcoin Core para iniciar un descubrimiento la primera vez. Ya que se ha realizado una conexión con la red se puede preguntar a ese nodo por la dirección de otros nodos.

Existen dos modos principales de operación de los nodos en la red:

Nodos completos

En este tipo de nodos se mantiene una copia local del blockchain. Es el modo empleado por Bitcoin Core. Cada nodo verifica de forma independiente las transacciones que son recibidas a través de la red P2P. Con este modo se asegura que se puedan verificar transacciones de forma independiente, sin tener que recurrir a fuentes externas. Los nodos completos dependen de la red para recibir actualizaciones acerca de los nuevos bloques de transacciones, los cuales verifican e incorporan a su copia local.

Nodos de verificación simplificada de pago

Debido a las restricciones de ciertos dispositivos, no siempre es posible el llevar una copia completa del blockchain. Para estos dispositivos se encuentra disponible el modo de *verificación simplificada de pago* (SPV, por sus siglas en inglés).

Los nodos SPV descargan únicamente los encabezados de los bloques en el blockchain. Al no poseer una lista completa de las UTXOs disponibles, los nodos SPV requieren de la ayuda de nodos completos para verificar que una transacción se encuentre en un bloque. A través de la raíz Merkle y una rama del arbol Merkle los clientes SPV pueden verificar si una transacción se encuentra en un bloque o no, así se conoce la profundidad dentro del blockchain en que se encuentra la transacción y el trabajo correspondiente que se requeriría para realizar un doble gasto. El cliente SPV posee las raíces Merkle así como la información de las transacciones de su interés, para verificar su inclusión solicita a un nodo completo la información de la rama Merkle.

No es posible engañar a un nodo SPV que una transacción existe en un bloque siendo que esta no existe realmente, pero lo contrario sí es posible. Por esto se recomienda que los clientes SPV se conecten a varios nodos al azar para minimizar la posibilidad de que los otros nodos le estén «escondiendo» información.

6.2.4. Minado

El minado es el proceso a través del cual se agregan nuevos bloques al blockchain, haciendo difícil de esta forma el poder modificar el historial de transacciones.

Los mineros verifican las transacciones que van llegando de la red y las agregan al blockchain. Un nuevo bloque de transacciones es minado aproximadamente cada 10 minutos. Todas las transacciones que se vuelven parte de un bloque el cual es agregado al blockchain se consideran confirmadas.

Los mineros reciben dos tipos de incentivos por su labor: se crean nuevas monedas en cada bloque y obtienen las comisiones incluidas en cada transacción. Para obtener estas recompensas los mineros deben de proporcionar una prueba de trabajo, la cual demuestra que el minero ha invertido una cantidad considerable de poder de cómputo al realizar esta labor.

El proceso de generación de nuevas monedas se denomina minado debido a que la recompensa es pequeña en comparación con el trabajo realizado, como sucede en la explotación de metales preciosos. En el inicio de Bitcoin en enero de 2009 se obtenía una recompensa de 50 Bitcoins por cada nuevo bloque. La cantidad de Bitcoins que se pueden obtener del proceso de minado disminuye a la mitad cada 210000 bloques (aproximadamente cada 4 años). Bitcoin fue diseñado con el concepto de tener una oferta monetaria finita, lo cual significa que sólo un número especifico de Bitcoins será creado. El limite superior de Bitcoins que existirán es de 21 millones, el cual se estima se obtendrá alrededor del año 2140. Cuando este número sea alcanzado

la única recompensa con que contarán los mineros serán las comisiones por procesar transacciones.

Aunque a través de la actividad del minado se puede obtener una ganancia significativa si se logra minar un bloque[9] la función principal de este proceso es asegurar las transacciones de la red y lograr un consenso sobre el estado de estas.

Altura del bloque y bifurcación

Cualquier minero que logre de forma satisfactoria obtener un picadillo del encabezado del bloque con un valor menor o igual al objetivo de la red puede agregar el bloque completo al blockchain.

A los bloques dentro del blockchain normalmente se les identifica por su altura: la distancia entre ellos y el primer bloque (bloque 0 o bloque origen).

Múltiples bloques pueden tener la misma altura, esto sucede cuando uno o más mineros producen un bloque nuevo al mismo tiempo. Esto crea una bifurcación aparente en el blockchain. Cuando esto sucede cada nodo selecciona uno de estos, normalmente el primero que reciben. Eventualmente, algún minero creará un nuevo bloque el cual extienda una de las bifurcaciones. Esto hace que ese lado de la bifurcación tenga mayor altura que los otros. Los nodos siempre siguen la cadena mas difícil de recrear y descartan los bloques relegados en bifurcaciones con menor altura (ver figura 6.10).

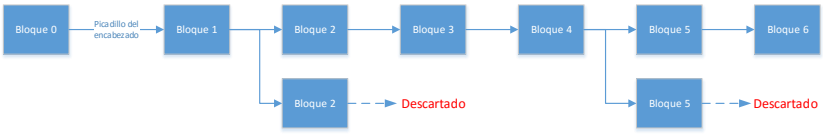


Figura 6.10: Bifurcación

Dado que multiples bloques pueden tener la misma altura cuando surge una bifurcación, la altura no debe ser usuada como un identificador global único. En su lugar se debe de emplear el picadillo del encabezado.

Transacción de acuñado

La primera transacción que se incluye en un bloque es una transacción especial, llamada *transacción de acuñado*. Esta es construida por el minero y es su recompensa por el trabajo realizado. En esta la estructura de la entrada difiere de una transacción normal (ver en la tabla 6.4).

Las transacciones de acuñado no tienen script de desbloqueo en su entrada, en su lugar se incluye información de acuñado la cual —a excepción de los primeros bytes donde se debe colocar la altura del bloque— es arbitraria y el minero puede colocar lo que desee. En la salida el minero coloca la suma de todas las comisiones más la recompensa por generar un nuevo bloque y lo asigna a su dirección.

La recompensa por generar un nuevo bloque se calcula de la siguiente manera:

[9] Aproximadamente el equivalente a 9400 dólares, calculado el 9 de noviembre de 2015.

Tamaño	Campo	Descripción
32 bytes	Picadillo de la transacción	Todos los bits en 0
4 bytes	Índice de la salida	Todos los bits en 1
1-9 bytes	Tamaño de la información de acuñado	Tamaño en bytes que tiene la información de acuñado. 2 a 100 bytes.
Variable	Información de acuñado	Información arbitraria empleada para números aleatorios adicionales e información de minado en bloques v2. Debe iniciar con la altura del bloque.
4 bytes	Número de secuencia	Valor 0xFFFFFFFF

Tabla 6.4: Entrada de una transacción de acuñado

Algoritmo Cálculo de recompensa

Entrada: Altura del bloque $nHeight$.
Salida: Recompensa $nSubsidy$.

1: $nSubsidy \leftarrow 50 * 100000000$
2: $halvings \leftarrow \lfloor nHeight/210000 \rfloor$
3: **if** $halvings >= 64$ **then**
4: **return** 0
5: **end if**
6: $nSubsidy \leftarrow nSubsidy/2^{halvings}$
7: **return** $nSubsidy$

Encabezado del bloque

El encabezado del bloque tiene un tamaño de 80 bytes y posee el formato descrito en la tabla 6.5. Cada vez que un minero quiere generar un nuevo bloque debe de rellenar todos los campos.

A lo largo de la evolución de Bitcoin han surgido diferentes versiones de las reglas que se deben de seguir para que un bloque se considere válido. Actualmente se emplea la versión 3.

Para calcular la raíz Merkle primero se coloca la transacción de acuñado como la primera hoja del árbol y posterior a esta se colocan todas las transacciones a incluir. Se debe tomar en cuenta que el tamaño máximo de un bloque es de 1MB.

El objetivo es un entero sin signo de 256 bits ante el cual el valor obtenido del picadillo del encabezado del bloque debe ser igual o menor para que pueda ser considerado como un nuevo bloque en el blockchain. Como se puede ver en la tabla 6.5 el campo *Objetivo* únicamente provee de 32 bits de espacio, por esto se emplea un formato menos preciso llamado «compacto» el cual funciona como una versión base 256 de la notación científica:

Bytes	Nombre	Tipo de dato	Descripción
4	Versión	uint32_t	La versión del bloque empleado, esto determina las reglas de validación que se deben de seguir
32	Picadillo del bloque previo	char[32]	Un picadillo SHA-256(SHA-256()) del encabezado del bloque previo. Su función es que ningún bloque se pueda modificar sin tener que modificar todos los bloques posteriores.
32	Raíz Merkle	char[32]	Un picadillo SHA-256(SHA-256()). La raíz Merkle se deriva de los picadillos de todas las transacciones incluidas en el bloque, asegurando de esta manera que ninguna transacción sea modificada sin tener que modificar el encabezado
4	Fecha	uint32_t	La fecha, en formato UNIX, cuando el bloque comenzó a ser minado (de acuerdo al minero). Debe ser mayor o igual a la mediana de los 11 bloques previos. Los nodos completos no aceptan encabezados con más de dos horas en el futuro, de acuerdo a su reloj.
4	Objetivo	uint32_t	El valor objetivo el cual el picadillo de este bloque debe ser menor o igual.
4	Número arbitrario	uint32_t	Un número arbitrario el cual los mineros modifican para poder producir un picadillo con valor menor o igual al objetivo. En el caso de que las 2^{32} posibilidades se hayan probado sin éxito, se puede modificar la fecha o las transacciones a incluir, modificando de esta manera la raíz Merkle

Tabla 6.5: Formato del encabezado

0x181bc330	→	0x1bc330	*	256	#	(0x18	-	3)
Valor del objetivo		Mantisa		Base		Exponente		Bytes en la mantisa

Resultado: 0x1bc330000000000000000000000000000000000000000000

El valor objetivo mínimo es 0x1d00ffff. A veces se emplea el término *dificultad*, esta se puede calcular de la siguiente manera:

$$\text{dificultad} = \text{objetivo mínimo}/\text{objetivo actual}$$

Nodos mineros

Algunos de los nodos en Bitcoin son nodos especializados llamados mineros. Estos al igual que los demás nodos reciben y propagan transacciones para ser agregadas al blockchain, pero también realizan la acción de generar bloques para ser agregados al blockchain. Ellos compiten para lograr agregar ese bloque al blockchain y con esto obtener su recompensa. Cada vez que se anuncia que un nuevo bloque ha sido creado, para los nodos mineros significa que ha habido un ganador. Ya que han verificado, agregado este bloque a su copia local del blockchain y lo han enviado a los demás, estos comienzan de nuevo a competir por generar el bloque siguiente.

Proceso de minado

El proceso de minado se puede resumir de la siguiente manera:

1. **Recepción y verificación de transacciones**: cada cliente que desea que su transacción sea incluida en el blockchain la envía a los nodos con los cuales está conectado. Cada nodo que recibe una transacción primero la verifica, si es válida la envía a su vez a los demás nodos con que está conectado.

2. **Creación de bloque candidato**: después de haber verificado las transacciones los nodos las almacenan temporalmente hasta que sean agregadas al blockchain. Los nodos mineros además de realizar estas acciones también agrupan las transacciones en un bloque candidato e insertan al principio una transacción de acuñado.

3. **Creación del encabezado del bloque**: ya que el nodo minero ha creado un bloque candidato, este genera el árbol Merkle y calcula su raíz. Con este dato ya puede generar el encabezado del bloque.

4. **Minado**: el minero realiza un picadillo SHA-256(SHA-256()) del encabezado del bloque y verifica si el valor obtenido es menor o igual al objetivo establecido por la red. En caso de que no sea así lo intenta de nuevo incrementando el valor del campo *numero arbitrario* en el encabezado del bloque. Si ya ha intentado todos los valor posibles del campo *numero arbitrario* y aún no logra obtener un valor menor o igual al objetivo entonces debe de modificar el encabezado de otra manera, normalmente se toma la estrategia de llevar un contador secundario en el área de datos de acuñado (lo cual modifica la raíz Merkle); aunque también es válido modificar la fecha y quitar o agregar transacciones al bloque candidato.

5. **Transmisión y verificación del nuevo bloque**: cuando un minero logra obtener un valor del picadillo del encabezado menor o igual al objetivo de la red, éste lo transmite a los demás nodos. Los demás nodos lo verifican y si cumple con todas las reglas lo incorporan a su copia local del blockchain.

6. **Confirmación del bloque**: al existir la posibilidad de bifurcaciones, el haber creado un bloque que cumpla con el objetivo no significa que este será definitivamente incorporado al blockchain, es necesario que se creen bloques adicionales que tomen como padre a este bloque.

¡Sé maduro!

Debido a la posibilidad de bifurcaciones, los satoshis obtenidos como recompensa en la generación de un nuevo bloque no pueden ser empleados hasta que cumplan con un nivel de madurez de 100 bloques. Esto quiere decir que desde el bloque más reciente al bloque en donde se encuentra la transacción de acuñado deben de existir por lo menos 100 bloques de diferencia.

6.2.5. Seguridad

Una de las principales características de Bitcoin es ser descentralizado, lo cual genera implicaciones de seguridad importantes. En un modelo centralizado, como un banco tradicional, se requiere de un estricto control de acceso para mantener fuera a los atacantes; sin embargo en Bitcoin toda la información es pública. A continuación se hace un recuento de varias implicaciones de seguridad que tiene el diseño y la operación de Bitcoin.

Criptografía

Para poder asegura las transacciones Bitcoin emplea de forma extensiva la criptografía de curva elíptica [Miller, 1986] [Koblitz, 1987]. Para firmar una transacción se emplea el estándar ECDSA [NIST, 2013] empleando los parámetros de la curva `SECP256k1` [SECG, 2009]. Esta curva tiene un orden primo de 256 bits. Los coeficientes de la curva son $a = 0$ y $b = 7$. Esto significa que `SECP256k1` tiene un j-invariante 0 y por lo tanto tiene una estructura muy especial.

Una curva con j-invariante 0 tiene endomorfismos computables de forma eficiente los cuales pueden ser usados para acelerar las implementaciones, por ejemplo utilizando la descomposición GLV para realizar la multiplicación escalar [Gallant et al., 2001]. Ya que en `SECP256k1` $p \equiv 1 \mod 6$ entonces existe una 6ta raíz de unidad $\zeta \in \mathbb{F}_p$ y un automorfismo $\psi : E \to E, (x, y) \mapsto (\zeta x, -y)$. Este mapa permite el cálculo rápido de ciertos múltiplos de cualquier punto $P \in E(\mathbb{F}_p)$, siendo $\psi(P) = \lambda P$ para un entero λ con $\lambda \equiv 1 \mod n$. Pero los endomorfismos eficientes no sólo aceleran el cálculo de la multiplicación escalar, también aceleran el cálculo del logaritmo discreto empleando el algoritmo de ρ de Pollar. El grupo del automorfismo de E tiene orden 6 y es generado por el mapa ψ descrito. En contraste, una curva elíptica con j-invariante diferente de 0 y 1728 sólo tiene grupo de automorfismo de orden 2, tal que la aceleración en el algoritmo ρ de Pollar es un factor constante de hasta $\sqrt{3}$ sobre tal curva [Bos et al., 2014].

En [Gilbert and Handschuh, 2004] se hace un estudio de la seguridad de SHA-256 contra ataques de colisión, se analizan las propiedades de seguridad de su estructura básica y se llega a la conclusión de que ataques diferenciales y lineales no aplican para esta.

Doble gasto

En los modelos centralizados de dinero electrónico con monedas identificables (por numero se serie) el problema del doble gasto es fácil de detectar: el emisor verifica si la moneda ya ha sido empleada. Al no tener Bitcoin un ente central que verifique si una moneda ha sido previamente usada, este se protege del doble gasto a través de la regla: sólo las salidas de transacciones previas sin gastar pueden ser empleadas como entrada en transacciones posteriores. Esta regla es aplicada por los nodos al propagar las transacciones y al momento de minar. El orden es determinado por la posición del bloque en el blockchain. Para prevenir un ataque Sybil [Douceur, 2002] Bitcoin limita al atacante al emplear una prueba de trabajo. En [Miller and LaViola Jr, 2014] los autores modelaron el protocolo de Bitcoin como un algoritmo de consenso Bizantino, donde muestran que sí se llega a un consenso. Bitcoin logra óptima resistencia Bizantina, también llamada resistencia $2f + 1$, bajo la premisa de comunicación síncrona e incluso en la presencia de adversarios. Esto significa que el sistema es seguro siempre y cuando los n nodos honestos prevalezcan ante f adversarios con $n > 2f + 1$. En el caso de Bitcoin la red resiste la presencia de adversarios controlando menos de la mitad del poder de cómputo.

Ataque Sybil

Los sistemas punto a punto normalmente emplean la redundancia para limitar su dependencia en nodos potencialmente hostiles. Si no se establece una identidad para cada nodo ya sea de forma explicita (a través de CAs) o implícita (como puede ser por su dirección IP) estos sistemas son propensos a sufrir un ataque Sybil, en el cual un pequeño número de entidades fabrica múltiples identidades falsas con el fin de obtener una participación desproporcionadamente grande en el sistema.

Una descripción genérica de cómo se podría llevar a cabo un ataque de doble gasto es la siguiente:

1. El atacante publica a la red una transacción, por ejemplo para pagar un producto.

2. De forma secreta el atacante mina una bifurcación la cual extiende el último bloque del blockchain, esta contiene una transacción la cuál se contrapone a la enviada en el paso 1. Una transacción en la cual el atacante transfiere la misma UTXO pero a una dirección que él controla.

3. El atacante espera hasta que el vendedor esté convencido (recibe un número de confirmaciones que él considera aceptables) y le entregue el producto.

4. Tan pronto la bifurcación que incrementa el tamaño del blockchain es mayor a la cadena actual esta se transmite a la red.

Dado que la bifurcación liberada por el atacante es mayor a la actual, la red la toma como la cadena principal. Invalidando de esta manera el pago realizado al vendedor.

Otra forma de lograr un doble gasto, pero en un escenario más es específico es la descrita en [Karame et al., 2012] donde se discute el entorno de transacciones rápidas, por ejemplo en una cafetería, donde el esperar mucho tiempo para obtener un número considerable de confirmaciones entorpecería o ocasionaría una gran pérdida de clientes. En esta situación, al tratarse normalmente de transacciones con un bajo valor se acepta hasta cierto punto el riesgo de un doble gasto ya que el implementar medidas más severas implicaría una pérdida mayor. En estas situaciones se recomienda tener una cantidad considerable de nodos auxiliares que verifiquen que una solicitud de transacción haya alcanzado a un número considerable de nodos, para evitar el siguiente ataque:

1. El atacante identifica al vendedor y los nodos con quien se conecta.

2. El atacante consigue que los nodos con los que se conecta el vendedor sean sus cómplices, así como que él mismo esté conectado con el vendedor y posee otros cómplices en la red.

3. Al pagar por un artículo, digamos un café, el atacante envía la transacción al vendedor, este la retransmite a los nodos con los que está conectado pero estos descartan la transacción.

4. El atacante envía a la red una transacción que se contraponga a la emitida al vendedor y se asegura de que sea difundida correctamente por la red.

5. Al llegar la transacción del atacante a una porción considerable de la red y la del vendedor ser desechada o transmitida de forma limitada, el atacante posee una ventaja para que su transacción sea la que acabe siendo minada y por lo tanto confirmada.

La mejor forma de prevenir un doble gasto es esperar varias confirmaciones —normalmente se toman 6 confirmaciones como un parámetro seguro—, aunque se pueden tomar previsiones adicionales como no aceptar conexiones directas o tener una conjunto de nodos confiables y distribuidos alrededor del mundo para verificar que las transacciones estén llegando a toda la red.

Aún cuando un ataque de doble gasto sea exitoso el carácter público del blockchain permite seguir el rastro de operaciones de doble gasto e identificar monedas que hayan estado implicadas [Gervais et al., 2014]. Es muy probable que las víctimas denuncien y sigan el flujo de esas monedas. Así otros vendedores podrían abstenerse de aceptar esas monedas o tomar medidas adicionales, como esperar un número mayor de confirmaciones, en el caso de aceptarlas.

Prueba de trabajo

El uso de una prueba de trabajo es un punto clave en Bitcoin. Cualquier tarea que sea apropiada para ser usada como una prueba de trabajo debe ser difícil de resolver pero fácil de verificar. Lo cual normalmente lleva a un proceso de búsqueda por fuerza bruta para intentar encontrar la solución.

Originalmente en Bitcoin el paradigma de la prueba de trabajo era «un CPU un voto» [Nakamoto, 2008]. Bitcoin usa una función basada en el desempeño del CPU (SHA-256) como prueba de trabajo. Los mineros por naturaleza buscan ganancias. Sus costos por realizar el minado constan del hardware y la energía consumida por este. Ellos buscan llegar al punto de equilibrio financiero lo más pronto posible, para poder obtener un margen de utilidad lo más grande posible. En un principio los mineros empleaban CPUs ordinarios para resolver la prueba de trabajo. Aunque los CPUs de propósito general son muy versátiles, eso lleva a una limitante en la velocidad. Por esto los mineros rápidamente empezaron a buscar soluciones más rápidas para ganarle a la competencia y tener mayores ganancias.

Las operaciones que se realizan en el proceso de minado son altamente paralelizables. Debido a esto se comenzaron a emplear procesadores gráficos (GPU), los cuales son capaces de calcular las múltiples operaciones de picadillos de forma más rápida y con mejor eficiencia energética. El uso de GPUs para minería pronto remplazó a los CPUs. Al popularizarse Bitcoin el nivel de competencia continuó incrementándose. El hecho de que las funciones basadas en el desempeño del CPU son susceptibles de ser aceleradas a través de hardware [Chaves et al., 2008] recibió atención. Era cuestión de tiempo para que las primeras soluciones de minado por hardware comenzaran a estar disponibles. Primero se comenzó con el uso de FPGAs[10] y posteriormente se pasó al uso de ASICs[11]. Desde entonces la única manera económicamente viable de realizar minado es a través de ASICs, los cuales pueden realizar billones de operaciones de picadillo por segundo.

El uso de hardware especializado incrementa la capacidad de voto por individuo. Estos desarrollos atentan contra el paradigma de prueba de trabajo y por lo tanto implican una amenaza. En particular, se reduce la base democrática al eliminar a los «pequeños» mineros. Como resultado la confianza en Bitcoin decrece. En Bitcoin es más probable que «los ricos se vuelvan más ricos» en [Kondor et al., 2014] se muestra que el patrimonio de los usuarios ricos se incrementa de forma más rápida que la de los usuarios con saldos bajos. Adicionalmente existe una tendencia alarmante en la cual el poder de un grupo pequeño de mineros excede en poder de cómputo de manera significativa al del resto de los mineros que participan [Gervais et al., 2014].

Privacidad

En contraste con los bancos tradicionales donde un tercero de confianza es quien mantiene la información de las transacciones para sí mismo, en Bitcoin esta información es pública. Las direcciones en el blockchain están pensadas para funcionar como seudónimos para evitar que el carácter público del historial de transacciones implique de forma directa el poder identificar a alguien. Para que esto sea cierto es necesario que una nueva dirección sea usada en cada transacción. Esto está implementado y activado de forma predeterminada en los clientes de Bitcoin, en los cuales el cambio de una transacción siempre es enviado a una dirección nueva.

Sin embargo como se explica en [Nakamoto, 2008] el empleo de transacciones con múltiples entradas, revelan que éstas pertenecen al mismo individuo. De hecho, dado que las transacciones en Bitcoin forman una cadena de firmas digitales, el gasto de monedas individuales puede ser rastreado públicamente [Reid and Harrigan, 2013]. Tomando en cuenta que toda la información de las transacciones es pública se ha

[10] Field Programmable Gate Array
[11] Application-Specific Integrated Circuit

realizado una gran cantidad de estudios sobre esta información como los realizados en [Ron and Shamir, 2014], [Ron and Shamir, 2013], [Reid and Harrigan, 2013], [Androulaki et al., 2012], [Meiklejohn et al., 2013], [Ober et al., 2013], [Baumann et al., 2014] y [Vasek and Moore, 2015].

La metodología que es normalmente empleada para el análisis del blockchain es la siguiente:

1. **Grafo de transacciones**: se crea un grafo $\mathcal{T}(T, L)$, donde T es el conjunto de transacciones en el blockchain y L es el conjunto de asignaciones directas (relaciones de entrada salida en transacciones) entre estas transacciones. Cada asignación $l \in L$ lleva un número de monedas C_l. De forma inherente las transacciones tienen un orden total definido por el blockchain, así que como se indica en [Reid and Harrigan, 2013] no pueden existir ciclos en $\mathcal{T}$.

2. **Grafo de direcciones**: de las asignaciones en el grafo de transacciones se pueden inferir los pares de direcciones origen-destino. Esto es posible para la mayoría de las transacciones estándar, como P2PKH. Basandose en estas relaciones se puede obtener el grafo $\alpha(A, L')$ donde A es el conjunto de direcciones de Bitcoin y L' es el conjunto de asignaciones directas, pero esta vez conectando direcciones en lugar de transacciones. Opcionalmente se puede transformar a α en un multigrafo al añadir la fecha como un atributo a cada $l \in L'$ para poder distinguir entre múltiples asignaciones entre un par de direcciones. Hay que tener en cuenta que algunas asignaciones no poseen un par de origen destino, por ejemplo las transacciones de acuñado.

3. **Grafo de entidades**: se agrupan las direcciones que probablemente pertenecen a un mismo dueño para construir un grafo de entidades. Esto se realiza a través de diferentes heurísticas basadas en el protocolo de Bitcoin y en prácticas comunes. El grafo de entidades $\epsilon(E, L'')$ consiste de un conjunto E de entidades, donde cada $\epsilon \in E$ es un subconjunto disjunto de direcciones A. La heurística más aceptada es el suponer que todas las direcciones de las entradas de una transacción le pertenecen a la misma entidad.

Para poder prevenir que el uso de técnicas de análisis del blockchain sean exitosas se requiere desacoplar la información del emisor y el receptor. Para esto existen estrategias de «lavado» o «mezclado» de bitcoins. La forma más fácil es a través de un tercero de confianza el cual recibe las monedas más una comisión y cuando se le solicita este las transfiere al destino especificado. Este tercero almacena en una única dirección todas las monedas recibidas y posteriormente las distribuye. De esta manera no es posible construir un camino con las transacciones.

El rastrear el flujo las transacciones no sólo ayuda a entender la red, también ayuda a obtener información de los usuarios. En [Kaminsky, 2011] señala que al controlar un nodo que se conecte con todos los demás nodos es posible conocer la dirección IP origen de las transacciones. Al conocer el origen una transacción normalmente se conoce el emisor de la misma. Esto hace inútil el uso de seudónimos .

El uso de servicios como Tor [Dingledine et al., 2004] provee una solución al problema de de privacidad al no revelar la dirección IP origen. Aunque en [Biryukov and Pustogarov, 2015] se indica una manera en la cual un ataque de negación de servicio podría ser dirigido hacia los nodos de salida de Tor para desconectar a estos de la red de Bitcoin.

Cliente

Lo primero que se debe de tener para poder usar Bitcoin es una programa de cartera. Cada cartera mantiene las llaves privadas de los usuarios, por lo tanto es esencial tomar medidas para protegerla. A diferencia

de diferentes aplicaciones de criptografía, los usuarios que pierden o cuyas llaves son comprometidas tiene una pérdida económica inmediata e irrevocable. Los desarrolladores del software de Bitcoin Core han intentado una variedad de soluciones para resolver o al menos hacer más transparentes los problemas de usabilidad que surgen con el manejo y almacenamiento de llaves. En [Eskandari et al., 2015] se propone una serie de criterios de evaluación para la usabilidad de las interfaces de administración de llaves y concluye que las herramientas actuales emplean metáforas complejas las cuales no capturan de forma completa las implicaciones de las acciones de administración de llaves.

Existen varias estrategias empleadas para la administración de llaves en Bitcoin [Bonneau et al., 2015]:

- **Almacenamiento en dispositivo**: almacenar las llaves de forma directa en el disco es la forma más simple, pero estas pueden ser robadas por malware específicamente diseñado para este propósito [Litke and Stewart, 2014]. Normalmente los clientes transfieren el cambio a una dirección nueva lo cual requiere de un nuevo respaldo del archivo, otras soluciones son enviar el cambio a la dirección origen y el emplear direcciones generadas a partir de una semilla.

- **Control dividido**: para evitar un único punto de falla y mejorar la seguridad se pueden emplear scripts de multifirma. Para ser liberados se requiere de la firma con k_i para $i = \{1, 2, 3, \ldots, n\}$ con n como el número mínimo de firmas especificadas. Un ejemplo puede ser una cartera que requiere de la firma tanto de la computadora personal como del teléfono inteligente del propietario para liberar una UTXO. Adicionalmente si se quieren mantener ocultas las llaves públicas y la cantidad de firmas requeridas para liberar la UTXO se puede recurrir al uso de P2SH.

- **Carteras protegidas por contraseña**: la cartera puede soportar la opción de cifrar el archivo donde se almacenan las llaves privadas, la llave empleada se deriva de una contraseña proporcionada por el usuario. La protección por contraseña puede ayudar en algunas situaciones de robo, al ser necesaria la contraseña el atacante necesita instalar software que almacene las teclas pulsadas por el usuario o intentar encontrar la contraseña.

- **Llaves derivadas**: las llaves privadas pueden ser generadas a partir de un único secreto proporcionado por el usuario. Esto permite la fácil interacción con varios dispositivos. El olvido del secreto implica la pérdida de todos los fondos.

- **Almacenamiento en frío**: constan de medios pasivos de almacenamiento como papel o una memoria. Proveen resistencia ante malware. Las llaves son descartadas una vez que se usan y se deben generar nuevas. Para el uso de papel se emplean códigos QR para facilitar su uso, aunque se debe tener cuidado de no dejar a la vista el código correspondiente a la llave privada ya que basta una imagen para poder obtenerla.

- **Equipos aislados o hardware dedicado**: Los equipos aislados son un caso particular del almacenamiento en frío, donde los dispositivos pueden hacer operaciones, como lo es realizar firmas. Los equipos aislados pueden prevenir ciertos tipos de ataques al nunca estar en comunicación con Internet. En este caso se debe también tomar en cuenta la seguridad física del dispositivo. Los dispositivos de hardware dedicados emulan a un equipo aislado al prevenir que el equipo anfitrión tenga acceso a las llaves y únicamente provee métodos para firmar transacciones.

- **Carteras en la nube**: existen servicios de terceros los cuales ofrecen todas las funcionalidades de una cartera completa pero a través de una página web. Los medios de autenticación son contraseñas y algunos incorporan mecanismos de dos factores de autenticación. Son muy fáciles de usar y no requieren

de prácticamente ningún conocimiento técnico del usuario. Su uso requiere de la confianza en el proveedor. Ha habido varios casos de empresas que ofrecen estos servicios que han sido comprometidas [Moore and Christin, 2013].

El emplear el algoritmo ECDSA en la firma de las transacciones, es necesario que la implementación seleccione un número aleatorio para cada firma, este debe de mantenerse en secreto y no debe de repetirse nunca. De no ser así es posible obtener la llave privada a través de la firma.

Los robos debido a sistemas comprometidos, software con errores o un uso incorrecto son los riesgos de seguridad más probables en Bitcoin [Tschorsch and Scheuermann, 2015].

6.2.6. Economía

¿Cómo es que algo intangible puede tener un valor? Bitcoin no se compone mas que de bits. A diferencia de los pesos los bitcoins no tienen una forma física, ningún gobierno los respalda y operan con una regulación meramente técnica.

Consideremos cuarto respuestas. Primera, el valor técnico de Bitcoin recae en solucionar el problema del doble gasto. Cada transacción emplea criptografía de llave pública para crear un registro público permanente de quién es el dueño de una moneda. Aunque alguien observe una transacción no es posible crear copias de las monedas. Solo el propietario puede gastarlas.

Segunda, la red de Bitcoin hace posible un comercio más ágil, ya que las comisiones son muy bajas. Las compañías de tarjetas de crédito cobran alrededor del 2 % al 4 % del valor de la transacción. Cualquier comerciante que este teniendo una utilidad del 5 % no quiere perder la mitad en pagarle al procesador de pagos. Al reducirse las comisiones la gente es más proclive a usar su dinero y la economía crece.

Tercera, Bitcoin es mejor que las tarjetas de crédito para detectar un fraude por que cada transacción requiere de la autorización del comprador y no se pueden crear cargos falsos.

Cuarta, por que la gente lo acepta. Igual que cualquier otra forma de dinero [Van Alstyne, 2014].

La crisis financiera ha propiciado la pérdida de confianza en muchos de los intermediarios financieros, plataformas de comercio y sistemas de pago. La mayor innovación de las criptomonedas es su capacidad de transacciones que no requieren de confianza (no se necesita de un tercero de confianza). El trueque siempre ha sido posible, pero es un medio de intercambio pobre y no siempre puede ser empleado como almacén de valor.

Las criptomonedas nunca podrán convertirse en una alternativa al dinero de curso legal, por la simple razón de que las personas debe pagar sus impuestos. Esto protege al dinero fiduciario ser desplazado y el miedo a una pérdida del control monetario no debe ser una argumento para prevenir que Bitcoin circule como una moneda en paralelo. Aunque la tecnología de protocolos de pagos digitales no debe ser confundida con el problema de una moneda paralela [Blundell-Wignall, 2014].

Volatilidad

Bitcoin ha tenido por lo menos cinco ajustes significativos en su precio desde 2011 [Lee, 2013]. Estos ajustes se parecen a las burbujas especulativas tradicionales: la cobertura mediática sobreoptimista de Bitcoin lleva a olas de inversores novatos a inflar los precios [Salmon, 2013]. La especulación llega hasta un punto crítico tras el cual el precio eventualmente se desploma. Los nuevos inversores interesados en participar

corren el riesgo de sobrevaluar la moneda y perder su dinero en un desplome. La fluctuación del valor de Bitcoin hace que algunos observadores sean escépticos sobre el futuro de la moneda [Brito and Castillo, 2013].

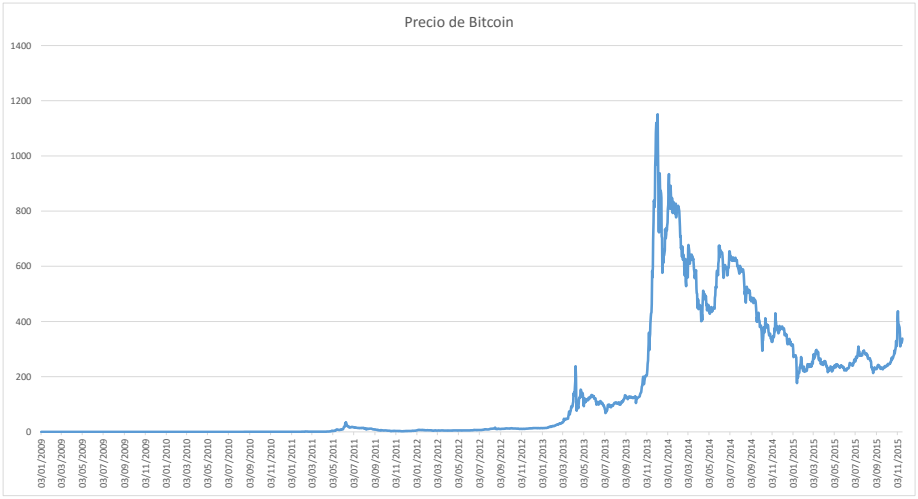


Figura 6.11: Precio de Bitcoin a lo largo del tiempo. Fuente: blockchain.info

En la figura 6.11 se muestra como ha variado el precio (en dólares) de un bitcoin a lo largo de su historia. En 2012 su precio sólo tenía un dígito, en la mayor parte de 2013 se mantuvo alrededor de los 100 dólares y después su precio se incremento rápidamente hasta los 1151 dólares al final del año para posteriormente colapsarse a al rango de los 500 a 800 dólares los primeros meses de 2014.

Los precios extremadamente altos pueden explicarse a través de una gran demanda inelástica y una oferta limitada. Suponiendo que el anonimato sea importante para algunos participantes del mercado para poder evadir impuestos o lavar dinero, la demanda puede exceder la oferta de minería. El lado de la oferta puede ser también un factor. Por ejemplo, la dificultad de minar puede de repente acelerarse o los mineros se asocian creando una asociación tipo cartel. De forma alternativa la demanda especulativa puede entrar en el mercado, con cada intermediario que cree que comprar a 10 o 20 veces el costo de minado no importa, siempre y cuando haya alguien dispuesto a pagar más (la teoría del más tonto). Juzgando por el repentino y extremo incremento en el volumen de compra venta cerca del incremento en el precio, la teoría del más tonto es la mejor explicación para el pico. La volatilidad del precio parece tener poco que ver con el valor justo y parte del problema en valuar criptomonedas es que esta es una tecnología y no un negocio con un reporte financiero o que estar respaldado por un bien físico [Blundell-Wignall, 2014].

Modelo deflacionario

Ya sea por accidente o por diseño Bitcoin como está especificado hoy en día define una moneda con características extremadamente deflacionarias. Las monedas son minadas por los verificadores como un incentivo para hacer que el ecosistema de Bitcoin se mantenga, pero el acuñado esta programado para expirar gradualmente y finalmente resultará en un límite en el total de bitcoins. Además las monedas para las cuales sus llaves privadas se pierdan, nunca podrán ser recuperadas, lo cual resulta en un decremento en la base monetaria. En perspectiva, de los 21 millones de monedas, 14.8 millones ya han sido minadas y de esos decenas de miles se reportan como perdidos.

Además de las consideraciones económicas, el potencial de una espiral deflacionaria en un sistema descentralizado como Bitcoin tiene implicaciones de seguridad que no deben de ser ignoradas.

Espiral deflacionaria

Con una oferta limitada Bitcoin no tiene otra alternativa mas que incementar su valor si es que el sistema gana un margen aceptable de popularidad. Incluso en el escenario de un «mercado maduro» con un 1 % de PIB de los Estado Unidos siendo operado en bitcoins y un 99 % en efectivo, el poder adquisitivo de las monedas se incrementaría con el tiempo, y cada moneda reflejaría en forma proporcional el crecimiento de la riqueza del país. Puesto de otra manera, mientras la Reserva Federal puede incrementar la cantidad de dólares circulando para acomodar el crecimiento económico, en Bitcoin la única manera sería el incremento en su valor. Mientras que se ha observado que una oferta monetaria fija puede llevar a espirales deflacionarias, es una paradoja que una de las fortalezas de Bitcoin pueda volverse su mayor debilidad, causando un final más catastrófico que una mera deflación.

Bitcoin más que cualquier otra moneda existente deriva su valor de la presencia de una infraestructura viva y dinámica acoplada de forma débil por una red de verificadores participando en la creación de bloques. Debido al incremento en su valor, los satoshis tenderían a ser guardados en vez de ser gastados. Esto crearía un decremento de la base en circulación, el volumen de transacciones disminuiría y la creación de bloques se volvería menos redituable (se colectarían menos comisiones). Si la circulación cae demasiado se puede precipitar una pérdida de interés en el sistema, resultando en la salida de mineros, hasta el momento en que el sistema se vuelva muy débil para protegerse [Barber et al., 2012].

6.2.7. Legislación

La visión original de Bitcoin esta ampliamente en tensión con las regulaciones y el control gubernamental. En este ámbito Bitcoin extiende una línea de ciberlibertad la cual data por lo menos hasta 1996 cuando John Perry Barlow emitió su «Declaración de la independencia del ciberespacio» [Barlow, 1996] en la cual rechaza el rol de los gobiernos en vigilar las comunicaciones en línea. Pero contrario a la visión inicial de Bitcoin en la que la descentralización lo hacía imposible de regular, ahora parece haber una amplia posibilidad de regulación así como circunstancias en las cuales tal intervención pueda ser de utilidad [Rainer Böhme, 2015].

Lucha contra el crimen

Bitcoin recibe escrutinio regulatorio por tres clases de preocupaciones criminales: crímenes específicos de Bitcoin, lavado de dinero y Bitcoin como facilitador para el crimen.

Los crímenes específicos de Bitcoin son ataques hacia la moneda o a su infraestructura, como lo son el robo de bitcoins, ataques a las *pools* de minado y ataques de negación de servicio hacia casas de cambio para manipular el precio. Las entidades de impartición de justicia normalmente tienen dificultades al prevenir o resolver estos crímenes debido a su novedad, falta de claridad de que agencia y jurisdicción son los responsables, complejidad técnica, incertidumbre de procesos y recursos limitados.

En el caso de que Bitcoin sea usado para lavado de dinero, esto propicia que los fondos sean más difíciles de rastrear, particularmente con el uso de mezcladoras. Estas características podrían ayudar al perpetrador a ocultar o enmascarar este crimen. Aunque Bitcoin también incluye en su diseño elementos los cuales podrían ayudar a rastrear fondos, incluyendo el carácter público del blockchain.

Finalmente, el aspecto de Bitcoin como facilitador de crímenes conlleva el pago de servicios ilegales los cuales son llevados acabo fuera de línea, como los artículos ilegales vendidos en el sitio Silk Road

[Ron and Shamir, 2014] y el pago de extorsiones. Los criminales podrían verse atraidos a Bitcoin debido a la falta de regulación, por la irreversibilidad de las transacciones o por que han sido bloqueados o expulsados de otros mecanismos de pago.

Protección al consumidor

Otra justificación para la regulación es la necesidad de protección al consumidor. Estas discusiones han sido particularmente más frecuentes después de la quiebra de la casa de cambio de bitcoins Mt. Gox en febrero de 2014, donde se perdió una cantidad de bitcoins valuada en más de 300 millones de dólares. A la luz de esta falla y otras más [Moore and Christin, 2013], es deseable tener procesos los cuales liquiden de forma equitativa los remanentes. El riesgo de colapso también requiere de la concientización del consumidor sobre los productos que está comprando.

Preocupaciones por una protección más amplia al consumidor resultan de la irreversibilidad de las transacciones en Bitcoin. La mayoría de los sistemas de pago proveen mecanismos para proteger a los consumidores de transferencias no autorizadas y tales protecciones son normalmente especificadas en la ley.

Legislación en México

En México únicamente existen dos antecedentes en los cuales entidades gubernamentales se han pronuncian sobre las operaciones con activos virtuales:

En marzo de 2014 el Banco de México emite el comunicado de prensa titulado: «Advertencias sobre el uso de activos virtuales como sucedáneos de los medios de pago en moneda de curso legal».

> Dado que en el entorno internacional ha aumentado el interés de los medios de comunicación y de ciertos sectores del público en los activos virtuales, tales como bitcoin, litecoin u otros similares, el Banco de México considera importante difundir información al respecto y alertar de los riesgos que conlleva el uso de dichos activos virtuales.
>
> Estos activos son mecanismos de almacenamiento e intercambio de información electrónica sin respaldo de institución alguna, por lo que no son una moneda de curso legal. El marco jurídico vigente tampoco los reconoce como medio de cambio oficial ni como depósito de valor u otra forma de inversión.
>
> Hasta ahora, los activos virtuales no han tenido en México una penetración relevante. Sin embargo, el Banco de México desea advertir al público respecto de los riesgos inherentes a la adquisición de estos activos y a su uso como sucedáneos de los medios de pago convencionales. Es importante resaltar que, además de lo aquí mencionado, otras autoridades podrían emitir consideraciones o regulación respecto de riesgos adicionales.
>
> En todo caso, estos activos presentan diferencias importantes con las monedas de curso legal, entre las que destacan:
>
> - No son monedas de curso legal en México, ya que el Banco de México no los emite ni respalda. De igual manera, tampoco son divisas extranjeras porque ninguna autoridad monetaria extranjera los emite ni respalda.
> - En consonancia con lo anterior, no tienen poder liberatorio de obligaciones de pago, por lo que su función como medio de pago no está garantizada toda vez que los comercios y demás personas no están obligados a aceptarlos.
> - El Banco de México no los regula ni supervisa.

- Las instituciones reguladas del sistema financiero mexicano no están autorizadas ni para usar ni para efectuar operaciones con ellos.
- En otras jurisdicciones, se ha señalado su uso en operaciones ilícitas, incluyendo aquellas relacionadas con fraude y con el lavado de dinero.
- No existe ningún tipo de garantía o regulación que asegure que los consumidores o comercios que adquieran este tipo de activos puedan recuperar su dinero. Más aún, al no existir una organización identificable que emita estos activos o un tercero que asuma obligaciones por dichos activos, difícilmente procedería un recurso legal en caso de pérdida.
- El precio en pesos mexicanos o en términos de otras monedas, determinado por las personas que aceptan comerciar con este activo, ha mostrado una gran volatilidad. Esto es consecuencia de su carácter altamente especulativo y de la elevada sensibilidad de su precio a cambios en la confianza de los usuarios (por ejemplo, cambios tecnológicos, surgimiento de nuevos activos virtuales, restricciones legales, etcétera). En consecuencia, la adquisición y el uso de estos activos conllevan un alto riesgo de depreciación y, por ende, de pérdidas monetarias.

Por lo anterior, cualquier persona que acepte este tipo de activo en intercambio de un bien o servicio, o lo adquiera, asume los riesgos arriba mencionados.

Si bien los activos virtuales actualmente no presentan un riesgo mayor para el sistema financiero ni para los sistemas de pagos, el Banco de México, en coordinación con otras autoridades, seguirá puntualmente su evolución y sus potenciales implicaciones y, de juzgarse necesario, emitirá la regulación pertinente.

En septiembre de 2015 la Secretaría de Hacienda y Crédito Público, en su Portal de Prevención de Lavado de Dinero emitió un aviso respecto a la utilización de activos virtuales en las operaciones establecidas en el Artículo 32 de la Ley Federal Para la Prevención e Identificación de Operaciones con Recursos de Procedencia Ilícita.

Conforme a las Directrices para un enfoque basado en riesgo para monedas virtuales, emitido por el organismo intergubernamental denominado Grupo de Acción Financiera Internacional (GAFI), el uso de activos virtuales a nivel internacional ha generado un nuevo método para la transmisión de valor a través del internet, por lo que se necesitan realizar acciones nacionales para identificar y mitigar cualquier riesgo de que dichos instrumentos sean utilizados en operaciones de lavado de dinero y financiamiento al terrorismo.

Por lo anterior y tomando en cuenta las facultades de esta Unidad Administrativa, se incluye dentro de la prohibición señalada en el artículo 32 de la LFPIORPI a los activos virtuales, de manera que queda prohibido dar cumplimiento a obligaciones y, en general, liquidar o pagar, así como aceptar la liquidación o el pago, de actos u operaciones mediante activos virtuales, en los supuestos señalados en dicho artículo.

Para tales efectos, se considera activo virtual, en singular o plural, al conjunto de datos almacenados en medios informáticos susceptibles de transmitirse electrónicamente que, sin ser moneda de curso legal en jurisdicción alguna, se utilizan como medio de cambio o unidad de cuenta para realizar operaciones de tipo comercial o económico o, en su caso, efectuar pagos.

No se considerarán activos virtuales a las unidades digitales utilizadas únicamente en plataformas de juego o como parte de programas de lealtad o recompensa a clientes, las cuales se utilizan sólo en compras relacionadas con el emisor o con los negocios afiliados a éste, pero no pueden ser convertidos por moneda de curso legal u otras divisas. Tampoco aplica dicha definición a las tarjetas de servicios, de crédito, de débito, prepagadas ni a otros instrumentos que constituyan instrumentos de almacenamiento de valor monetario correspondiente a una cantidad de dinero determinada, sean o no por Entidades Financieras.

6.2.8. Protocolos derivados

Bitcoin es un proyecto de código abierto y su código ha sido empleado como la base de otros proyectos. Los más comunes son del tipo *altcoin* los cuales emplean la base de Bitcoin para crear una nueva moneda.

También existen varias implementaciones que emplean como base a Bitcoin y su blockchain, trabajando como capas superiores a Bitcoin y su protocolo. Estas *metamonedas*, *metacadenas* o *aplicaciones de blockchain* usan el blockchain como una plataforma de aplicaciones o extienden el protocolo de Bitcoin al agregar capas encima.

Además de las altcoins también existen un número de implementaciones de blockchain alternativos, los cuales no son realmente monedas. Estos, denominados *altchain* implementan un algoritmo de consenso y una bitácora distribuida como una plataforma para contratos, registro de nombres y otras aplicaciones. Las altchains usan los mismos bloques básicos y algunas veces una moneda o ficha como medio de pago, pero su principal propósito no es el de fungir como moneda [Antonopoulos, 2014].

Metamonedas

Las metamonedas y metacadenas son capas de software implementadas encima de Bitcoin, ya sea implementando una moneda dentro de una moneda o una plataforma como una capa superior al protocolo de Bitcoin. Estas capas extienden las funcionalidades y el protocolo de Bitcoin al almacenar datos adicionales en las transacciones y direcciones de Bitcoin. Las primeras implementaciones de metamonedas empleaban varias estrategias para añadir metadatos en el blockchain de Bitcoin, estas eran el emplear campos no usados por Bitcoin (como el número de secuencia) o emplear las direcciones. A partir de la inclusión de la instrucción `OP_RETURN` las metamonedas han podido incluir información de forma más directa en el blockchain y la mayoría están migrando a su uso.

Un ejemplo son las *monedas de colores* estas constan de un meta protocolo que incluye información en una pequeña cantidad de satoshis. Una moneda de color es una cantidad de satoshis que ha sido convertida para representar otro bien. Imaginemos que alguien toma un billete de 100 pesos y le escribe «Esta es una acción de la empresa Patito S.A». Ahora el billete sirve para dos propósitos es un billete y una acción. Dado que su valor como acción es mayor que como billete el portador preferiría guardarlo que comprar algo con él. Las monedas de color funcionan de la misma manera, al convertir una pequeña cantidad de satoshis en un certificado que representa otro bien. El término de «color» alude a la idea de que se le da un significado especial a través de la adición de un atributo como lo es un color.

Altcoins

La mayor parte de las altcoins se derivan del código fuente de Bitcoin. La primera en ser anunciada fue IXCoin en agosto de 2011. IXCoin modificaba algunos de los parámetros de Bitcoin, específicamente incrementaba la recompensa a 96 monedas por bloque.

En septiembre de 2011 se lanzó Tenebrix. Esta fue la primera criptomoneda en implementar una prueba de trabajo alterna basada en el algoritmo *scrypt* el cual fue diseñado para realizar picadillos de contraseñas y una de sus propiedades es ser resistente a ataques de fuerza bruta. Uno de los propósitos de Tenebrix era el hacer difícil el uso de GPUs y ASICs al momento de minar empleando un algoritmo que hiciera uso intensivo de la memoria. Tenebrix no tuvo éxito como moneda, pero sirvió como la base de Litecoin la cual sí ha tenido gran aceptación y ha generado una gran cantidad de clones.

Litecoin además de emplear scrypt como algoritmo de prueba de trabajo también emplea un tiempo menor de generación de los bloques el cual es de 2.5 minutos. Debido a esto muchos piensan que Litecoin es más apto para ser usado en operaciones de venta al por menor, esto por que las confirmaciones son más rápidas que en Bitcoin.

Existen tres areas principales en las cuales las altcoins se diferencian de Bitcoin:

- Diferente política monetaria.
- Diferente prueba de trabajo o mecanismo de consenso.
- Características especificas, como mayor anonimato.

Altchains

Las altchains son implementaciones alternas del blockchain, las cuales no tienen como objetivo principal su uso como moneda. Muchas incluyen monedas pero las emplean como fichas para guardar algo, como un contrato o un recurso.

Namecoin fue una de las primeras implementaciones derivadas de Bitcoin. Namecoin es un sistema de registro de nombres el cual emplea como plataforma un blockchain. Este puede ser empleado como una alternativa a los servidores de nombre de dominio (DNS) el cual emplea el dominio `.bit`.

Etherum es otro ejemplo importante el cual es un sistema de procesamiento de contratos y ejecución basados en un blockchain. Etehrum emplea un lenguaje que es Turing completo. Tiene una moneda propia llamada *ether* la cual se emplea al momento de la ejecución.

Etherum puede implementar sistemas complejos los cuales son por si mismos un altchain. Por lo tanto Etherum es una palataforma para construir altchains.

Minado unificado

Las altcoins y altchains pueden aprovechar la popularidad de Bitcoin al emplear la misma prueba de trabajo. De esta manera un minero puede obtener varias monedas al precio de una. Para poder emplear esta estrategia la altcoin debe de ser compatible con minado unificado. El minado unificado aprovecha el espacio disponible en la entrada de la transacción de acuñado de Bitcoin al almacenar la información de la altcoin en este.

6.3. Ejemplo de uso de Bitcoin

Durante la realización del presente trabajo se empleó la criptomoneda Bitcoin como opción de pago para la Cuarta Conferencia Internacional en Criptología y Seguridad de la Información en América Latina —Latincrypt 2015—. Esta fue ofrecida como alternativa al pago a través de tarjeta de crédito. Para esto se diseñó e implementó una aplicación web para el registro y pago de los participantes. Ésta fue desarrollada en PHP.

6.3.1. Arquitectura

La arquitectura implementada es la mostrada en la figura 6.12, donde se puede apreciar que se emplea el concepto de defensa en profundidad. Toda comunicación proveniente de Internet es mediada por el cortafuegos, el cual sólo permite el uso de puertos autorizados (80 y 443). Posteriormente se emplea traducción de direcciones IP para pasar la comunicación al servidor proxy el cual verifica que la URL sea válida, en caso afirmativo la comunicación se entrega al servidor web, en el caso de no ser válida se despliega un mensaje de error. Finalmente, en caso de que el cliente desee pagar empleando Bitcoin se solicita una dirección nueva al servidor de Bitcoin, el cual únicamente puede comunicarse con el servidor web, se calcula el precio en bitcoins y se le presenta esta información al cliente. Si el cliente desea proceder esta información es almacenada en la base de datos y se le envía al cliente a través de correo electrónico.

Figura 6.12: Arquitectura del sistema de pago implementado

6.3.2. Asignación de direcciones

Las direcciones eran pregeneradas por Bitcoin Core, se contaba con un ciento de direcciones generadas. Cada que un cliente se registraba e indicaba que quería hacer su pago con bitcoins, el servidor web solicitaba una dirección disponible al servidor de Bitcoin, esta dirección era almacenada en la base de datos de la aplicación, posteriormente al finalizar el pedido se le desplegaba al usuario como un código QR y se le enviaba a su correo electrónico.

6.3.3. Tipo de cambio

Para calcular el tipo de cambio se empleó el API del sitio `www.deskcoin.com`, el cual provee de forma gratuita la información de los precios de diferentes casas de cambio (ver figura 6.13). Para realizar la conversión de pesos Mexicanos a bitcoins se tomaba el promedio del precio de cierre de los tres días previos. El cliente tenía un máximo de tres días para realizar el pago, en caso de no realizarlo se descartaba su solicitud y debía volver a hacer la orden de compra para obtener el precio actualizado.

6.3.4. Base de datos

Se empleó la de base de datos MariaDB 10.0. El esquema creado se puede consultar en el apendice B.

```
{
    "bpi": {
        "2015-11-01": 5363.1834,
        "2015-11-02": 5912.6356,
        "2015-11-03": 6579.3369
    },
    "disclaimer": "This data was produced from the CoinDesk
    Bitcoin Price Index. BPI value data returned as MXN.",
    "time": {
        "updated": "Nov 4, 2015 00:03:00 UTC",
        "updatedISO": "2015-11-04T00:03:00+00:00"
    }
}
```


Figura 6.13: Respuesta del precio de cierre de tres días

CAPÍTULO 7

PLANTEAMIENTO

Get your facts first, then you can distort them as you please.

—MARK TWAIN

X.509 ha evolucionado lentamente hasta convertirse en un modelo de infraestructura de llave pública extremadamente flexible. El problema con esto es que X.509 trata de ser todo a la vez, un enfoque tal vez práctico, pero como dice el dicho «el diablo está en los detalles».

X.509 fue originalmente diseñado para controlar el acceso a un directorio X.500, motivo por el cual no se contemplaron muchos de los problemas actuales. Esto crea una severa discordancia, ya que los requerimientos de las empresas deben ser ajustados a la fuerza para entrar en el modelo de X.509 para trabajar con certificados.

El modelo de X.509 para directorios, estructuras jerárquicas, revocación fuera de línea y otros aspectos que complican su uso surgen de su herencia de X.500. Idealmente el modelo debería emplear las herramientas y metodologías actuales, como lo son las bases de datos relacionales, una organización no jerárquica, así como validación y autorización en línea.

La solución a estos problemas es adaptar el diseño de la infraestructura de llave pública al mundo real, en vez de tratar de adaptar el mundo real a la infraestructura de llave pública. Existen una variedad de acercamientos alternativos, los cuales van desde simples parches hasta propuestas las cuales pueden ayudar a resolver los problemas inherentes al modelo del estándar X.509 [Gutman, 2002].

7.1. Certificados raíz

Los modelos de negocio de PKI que han evolucionado a lo largo del tiempo han tendido a encasillar usos particulares de esta tecnología, teniendo un gran énfasis la presencia de llaves públicas «raíz» (contenidas en un certificado autofirmado, llamado certificado raíz) en navegadores y sistemas operativos. Actualmente, las empresas que son dueñas de éstas —las autoridades certificadoras— cobran la emisión de certificados de llave pública anualmente. Los precios van desde los 49 USD hasta los 1699 USD[1]. Esta situación no funciona muy bien para sitios no comerciales. El conjunto de certificados raíz integrados en los navegadores y sistemas operativos también han dificultado la creación de nuevos nuevos servicios de PKI, los desarrolladores son casi forzados a emplear TLS o HTTPS debido a que de esta manera los navegadores y sistemas operativos confiarán en sus aplicaciones debido a los certificados preinstalados.

El almacenar un gran conjunto de certificados raíz en los navegadores y sistemas operativos tiene una debilidad potencial muy importante: el cliente confía ciegamente en todas las llaves públicas raíz, por lo que se confía inmediatamente en cualquier certificado que se emita con las mismas, no importando a nombre de quien se emita el certificado. Si alguna autoridad certificadora es comprometida o emite un certificado de manera incorrecta puede afectar a cualquier sitio y cliente en Internet [Farrell, 2011].

7.2. Ataques y malos manejos documentados

La seguridad es como una cadena, es tan fuerte como su eslabón más débil. La seguridad de los sistemas basados en autoridades certificadoras están compuesta por muchos eslabones los cuales no son únicamente criptográficos. Existe gente involucrada, esta puede actuar de manera maliciosa o descuidada comprometiendo el sistema completo. También los sistemas que emiten los certificados no pueden ser perfectos, siempre existirá una falla o vulnerabilidad potencial. Prueba de esto son los casos presentados a continuación, estos son una recopilación de los incidentes más importantes de los últimos cinco años con respecto a ataques y malos manejos que han sufrido varias autoridades certificadoras.

- **Diginotar**: Diginotar B.V. fue una empresa danesa fundada en 1998, la cual ofrecía servicios de certificación y poseía varias autoridades certificadoras. Los certificados emitidos por Diginotar eran válidos a nivel mundial. Los tipos de certificados emitidos eran: SSL (para sitios web), certificados calificados de acreditación (equivalente legal de una firma manuscrita) y certificados PKIoverheid (empleados en gobierno electrónico de Paises Bajos). En junio y julio de 2011 se lanzó contra la empresa un ataque exitoso, en el cual se lograron emitir 531 certificados apócrifos, los cuales fueron empleados desde agosto de 2011. Uno de los certificados emitidos fue a nombre de `*.google.com`, el cual se empleó para ataques de intruso de en medio principalmente en Irán [Fox-IT, 2011]. En septiembre de 2011 Digino-

[1] Precios actualizados al 19 de noviembre de 2015. Los precios corresponden a los productos Comodo Positive SSL y Symantec Secure Site Pro with EV, ambos con vigencia de un año y para un dominio. Adicionalmente, Symantec limita el uso del certificado a un equipo.

tar se declara en bancarrota. Los certificados raíz fueron eliminados en actualizaciones de los sistemas operativos y navegadores.

- **Comodo**: en marzo de 2011 una agencia registradora de la empresa Comodo fue atacada y como consecuencia una cuenta de usuario fue obtenida. Con la cuenta se generaron 9 certificados apócrifos de los cuales el emitido a nombre de `login.yahoo.com` registró actividad maliciosa. Los certificados fueron revocados posteriormente [Comodo, 2011].
- **ANSSI**: en diciembre de 2013 una autoridad intermedia de la empresa francesa ANSSI integró su certificado en un dispositivo comercial de inspección de trafico, acción que no es permitida por las CA raíz. El certificado fué revocado y Google limitó en su navegador los dominios para los cuales se confía en los certificados emitidos por ANSSI [Langley, 2013].
- **CNNIC**: en marzo de 2015 la empresa MCS Holdings, la cual es una autoridad intermedia de China Internet Network Information Center (CNNIC), empleó un certificado intermedio para generar certificados en un producto que se utilizaba para inspeccionar el tráfico en empresas. CNNIC revocó el certificado intermedio. Google, Microsoft y Mozilla tomaron varias medidas en sus navegadores [Langley, 2015].
- **live.fi**: en marzo de 2015, debido a una mala configuración por parte de Microsoft Finlandia se emitió un certificado a nombre de `live.fi`. Este fue revocado posteriormente [Microsoft, 2015].
- **Symantec**: el 14 de septiembre de 2015 Google detectó en las bitácoras de transparencia para certificados EV de Thawte, una subsidiaria de Symantec Corporation, la emisión de dos precertificados emitidos a nombre de `google.com` y de `www.google.com`, los cuales nunca fueron solicitados ni aprobados por Google. Tras hacer una solicitud de aclaración a Symantec se llegó a la conclusión de que los certificados eran parte de pruebas internas de la CA. Los certificados fueron revocados en el navegador Chrome y se informó que no se consideraba que hubiera habido algun riesgo para los usuarios [Somogyi and Eijdenberg, 2015]. Tras este incidente Symantec publicó un reporte [Symantec, 2015c] en el cual indicaba que tras su investigación había identificado que 23 certificados adicionales fueron emitidos sin permiso de los dueños de los dominios, entre estos se encontraban Google y Opera. A partir de este reporte Google revisó las bitácoras de transparencia y reportó haber encontrado certificados adicionales. Finalmente, tras una segunda auditoria por parte de Symantec se reportó que se encontraron 164 certificados adicionales correspondientes a 76 dominios [Symantec, 2015a] y 2458 certificados emitidos a dominios no existentes [Symantec, 2015b]. Debido a esto Google fijó a Symantec el 1 de junio de 2016 como fecha límite para implementar varias medidas que evitaran que esta situación volviera a suceder. En caso de que Symantec no las cumpla Google dará avisos, entre otras medidas, en sus productos cada que un certificado de Symantec se use [Sleevi, 2015].

7.3. Propuesta

¿Cómo es que Bitcoin elimina la necesidad de un banco central?

Bitcoin lo resuelve de una forma muy pragmática: en cierta manera, todos son el banco. Esto es cada participante posee una copia del registro que normalmente tendría el banco central —el blockchain—.

Si esto se puede lograr para crear una criptomoneda descentralizada, es muy razonable el pensar que esta estrategia pueda funcionar en el caso de la infraestructura de llave pública.

La propuesta es el aplicar los conceptos que emplea Bitcoin para asegurar sus transacciones, al campo de los sistemas de infraestructura de llave pública, teniendo como finalidad eliminar la necesidad de autoridades certificadoras y así lograr un sistema que opere de forma descentralizada.

Para lograr esto se realizó un diseño el cual emplea un blockchain para realizar lo que plantearon hace casi 40 años Diffie y Hellman [Diffie and Hellman, 1976]: un directorio público. Este con la característica de conceder el permiso de lectura a cualquiera y con un proceso de escritura realizado por una red punto a punto compitiendo por crear nuevos bloques. Aumentando con cada bloque la dificultad de poder modificar maliciosamente los datos ya existentes.

Al eliminar la necesidad de confianza ciega en una autoridad certificadora se logra una total transparencia en la emisión de certificados. También se disemina el riesgo de compromiso, ya que no existe una entidad central que emita los certificados.

7.3.1. Propuestas previas

Algunas de las propuestas existentes más interesantes son las siguientes:

- ✓ En [Garman et al., 2013], se expone la idea de emplear un registro público para la emisión y publicación de credenciales anónimas, eliminando de esta forma la necesidad de un emisor central en el que se deba confiar. Para esto, proponen emplear el protocolo de Namecoin, un derivado de Bitcoin empleado como alternativa al actual sistema de nombres de dominio. También realizan una implementación de su propuesta y dan un ejemplo de uso en certificación anónima distribuida directa, la cual emplea módulos de plataforma de confianza (TPM) en el lado del cliente para las operaciones realizadas.
- ✓ En [Fromknecht et al., 2014], se da la propuesta de emplear también el protocolo de Namecoin con algunas modificaciones para realizar ciertas funciones de una infraestructura de llave pública, principalmente enfocado a la distribución de llaves para cifrado de correo electrónico, pero en su diseño no se contemplan algunos puntos importantes como:
  - ✗ Posibles ataques de negación de servicio.
  - ✗ Verificación de la identidad del solicitante.
  - ✗ Purga de los certificados vencidos.
- ✓ Proyecto *Let's Encrypt* [(ISRG), 2014]: es un proyecto auspiciado por empresas y fundaciones como Cisco, Akamai, Electronic Frontier Foundation y Mozilla, entre otras. El objetivo es proporcionar una autoridad certificadora gratuita, fácil de emplear por los usuarios, transparente y cooperativa. La motivación principal de esta iniciativa es forzar a las autoridades certificadoras comerciales a transparentar su operación, así como a emplear estándares criptográficos más seguros. El inicio de operación del proyecto en fase de prueba abierta al público está planeada para el 3 de diciembre de 2015.

7.3.2. Requerimientos

Para que el sistema propuesto sea funcional debe tener las siguientes propiedades:

- **Descentralizado**: no debe existir un ente que concentre el poder y responsabilidad de emitir certificados. Esta tarea debe de ser distribuida entre todos los participantes.

- **Distribuido**: se emplea una red punto a punto (P2P) donde los procesos son totalmente descentralizados, funcionan como clientes y servidores a la vez. Todos los nodos tienen una copia de la información.
- **Auditable**: es fácil examinar, verificar o demostrar si un nodo intenta hacer trampa. Cualquier nodo de la red P2P puede realizar esta operación de forma independiente.

7.3.3. Propiedades

Las propiedades definidas para el sistema son:

- Blockchain 3.0
- Cadena alterna
- Enfoque en HTTPS
- Verificación a partir de DNS
- Compatible con X.509

Cada una de estas propiedades será explicada a continuación.

Blockchain 3.0

Los beneficios a los sistemas económico, político, humanitario y legal que brinda Bitcoin y la tecnología de blockchain empiezan a hacer claro que ésta es un tecnología con un potencial extremadamente disruptivo que puede tener la capacidad de remodelar todos los aspectos de la sociedad y sus operaciones. Por motivos de organización y conveniencia, los diferentes tipos de actividades existentes y potenciales en la revolución del blockchain se dividen en tres categorías [Swan, 2015]:

- **Blockchain 1.0**: son monedas, el desarrollo de criptomonedas en aplicaciones relacionadas a actividades monetarias como lo son transacciones, sistemas de pago y remesas.
- **Blockchain 2.0**: son contratos, el abanico completo de aplicaciones de mercado, económicas y financieras que emplean el blockchain y van más allá de simples transacciones de efectivo, como lo son acciones, bonos, opciones de compra, hipotecas, prestamos, títulos, propiedades y contratos inteligentes.
- **Blockchain 3.0**: son aplicaciones más allá de lo monetario, económico o financiero. Particularmente en áreas de gobierno, salud, ciencia, cultura y arte.

De estas categorías la que aplica al sistema propuesto es la de blockchain 3.0, ya que no se busca una aplicación con fines monetarios, económicos o financieros; además la aplicación se puede considerar dentro del sector de ciencia.

Cadena alterna

Una cadena alterna (altchain) es un sistema que emplea el algoritmo de encadenamiento de bloques para llegar a un consenso distribuido. Las altchains pueden compartir mineros con una red padre como lo puede ser Bitcoin, a esto se le conoce como minado unificado.

Bitcoin emplea el algoritmo de encadenamiento de bloques para llegar a un consenso acerca de quién es el dueño de las monedas. Las cadenas de bloques fueron inventadas específicamente para el proyecto de Bitcoin, pero pueden ser aplicadas en cualquier situación donde se requiera de un consenso distribuido en la presencia de actores malicioso o en los cuales no se confía.

Aunque es posible emplear el blockchain de Bitcoin para otros propósitos, para el diseño propuesto se decidió crear una red y un blockchain alterno debido a las siguientes razones:

- El blockchain de Bitcoin es una estructura de datos compleja y muy extensa. Verificar su integridad requiere de varias operaciones costosas y múltiples accesos a disco.
- Bitcoin no ha implementado hasta la fecha algún mecanismo de purga para evitar que el blockchain contenga información la cual ya no es relevante pero continúa usando espacio en cada uno de los clientes.
- Se tiene la flexibilidad de crear una estructura la cual cumpla cabalmente con los propósitos de nuestra aplicación. Empleando el blockchain de Bitcoin habría que idear estrategias las cuales permitieran trabajar con el modelo ya definido, además se está sujeto a los cambios que se realicen en Bitcoin.

Enfoque en HTTPS

Se escogió el protocolo de capa de aplicación HTTPS debido a que es el protocolo que se usa para consultar cualquier página web segura. Por lo que tiene un gran rango de aplicaciones como lo son servicios de banca en línea, consulta de web mail, comercio electrónico, redes sociales, etc. Además de que se apoya de gran manera en la jerarquía de nombres establecida por el sistema de nombres de dominio lo cual puede ayudar a extender fácilmente su uso a aplicaciones como lo es firma de correo electrónico.

Verificación a partir de DNS

Al estar el sistema enfocado a proteger comunicaciones a través del protocolo HTTPS es necesario verificar de alguna manera que quien trate de incorporar una llave al blockchain a nombre de un dominio realmente sea alguien autorizado para realizar esta acción. Esto con tal de prevenir posibles ataques de intruso de en medio o prácticas parecidas a la *ciberocupación*, en donde alguien registre una llave pública a nombre de un dominio para posteriormente tratar de obtener una ganancia económica al vendérsela a su legítimo dueño.

Para realizar esta verificación en la actualidad las autoridades certificadoras emplean diferente estrategias como lo son, consulta de registros WHOIS, comunicación a través de teléfono o correo electrónico con el administrador registrado del dominio, solicitud de documentos o solicitud de un cambio en el sitio web que emplea el dominio [CAB, 2015].

En el diseño propuesto se emplea la verificación de la pertenencia de un dominio a través de una consulta DNS. Donde el administrador de la zona coloca en un registro TXT la información de la llave que quiere registrar. Los registros TXT permiten almacenar hasta 255 caracteres en una cadena y a través del uso de varias cadenas es posible tener más espacio [ISC, 2011]. 255 caracteres es suficiente para poder publicar una llave pública empleando criptografía de curvas elípticas que sea soportada por TLS [Blake-Wilson et al., 2006].

El porqué de esta decisión es que se requiere de una forma automatizada de verificación y que sea fácil de implementar por parte de los administradores de la zona. Otra opción viable es la creación de un recurso

web para su consulta por los nodos de la red P2P, aunque esto podría llevar a un posible ataque de denegación de servicio ya que muchos equipos tratarían de acceder al mismo recurso en un tiempo corto, saturando el medio de comunicación y creando un pico en el procesamiento del servidor web. Además del uso cotidiano de *reverse proxies* o balanceadores de carga por sitios de tamaño mediano y grande, donde el *proxy* o el balanceador es la entrada al sitio, y es normalmente el punto donde se instalan los certificados, para posteriormente redirigir el tráfico a un servidor web. En empresas grandes el administrador de los *proxies* o balanceadores de carga no es el mismo del servidor web lo cual complica el proceso.

En el caso de DNS el escenario de un posible ataque de denegación de servicio es menos probarle debido a las diferentes jerarquías de los servidores en DNS y a los sistemas de *cache* que estos implementan.

Compatible con X.509

X.509 es el estándar *de facto* a nivel empresarial y comercial para el manejo de certificados. Todos los certificados que los sitios web usan siguen este estándar. Debido a esto es necesario poder interactuar con X.509, de no ser así es casi seguro que la solución propuesta no sería de interés para la industria. Esto debido a que se requeriría de un cambio sustancial en los productos existentes y se tendría un gran problema de incompatibilidad con sistemas heredados. Además que X.509 es una tecnología probada y que la mayoría de los administradores de sistemas conocen. Al tratar de imponer un nuevo esquema se tiene el riesgo de crear nuevas vulnerabilidades lo cual opacaría el beneficio potencial.

7.3.4. Operaciones

Las operaciones definidas para el sistema son las siguientes:

1. Alta
2. Baja
3. Revocación
4. Renovación

En las secciones subsecuentes se dará una descripción detallada de cada una.

Alta

La operación de alta solicita la inclusión de una nueva llave en el blockchain. Los pasos de esta operación son los siguientes:

1. Se genera un par de llaves. Este paso puede ser realizado en un equipo sin conexión con la red P2P.
2. El administrador de la zona coloca la llave pública en un registro TXT con el nombre `_dpkinpk` y como cadena coloca la llave pública en codificación Base58Check.
3. Se genera y transmite a la red una transacción `OP_ADDPK` que emplea como dirección el picadillo de la llave pública.
4. Cada nodo que recibe la transacción verifica que esté bien formada y coteja contra el servidor DNS la llave pública. En caso de coincidir se agrega a la cola para ser minada y se transmite a los demás nodos. En caso contrario es descartada.

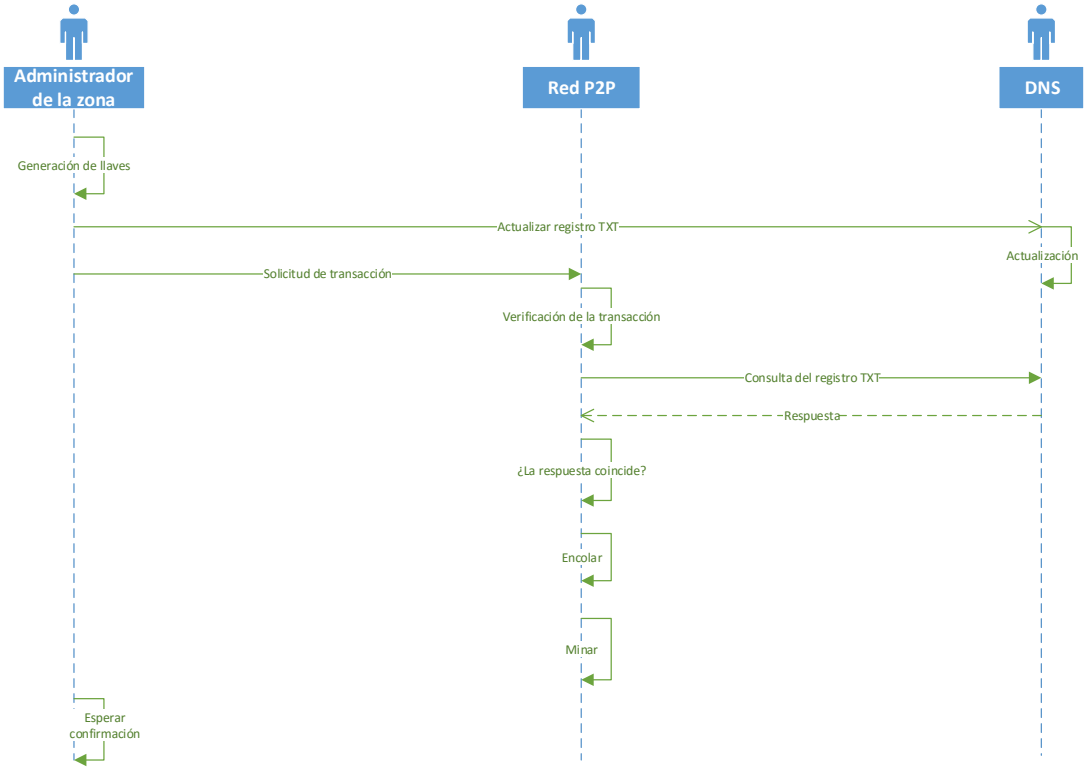


Figura 7.1: Diagrama del proceso de alta de una llave pública

5. Un nodo minero la agrega a un nuevo bloque.

6. Se debe esperar un a cantidad de bloques c para que la llave sea tomada como definitiva.

En la figura 7.1 se muestra un diagrama del proceso.

Baja

La operación de baja elimina una llave pública del blockchain.

Esta operación es realizada de forma automática cuando el bloque donde se agregó la llave pública ha sido confirmado un número determinado de veces. Esto es, existe una cantidad n de bloques que se han creado a partir de este bloque.

Un cliente no puede solicitar la baja de su llave pública, este debe esperar a que sea automáticamente dada de baja.

Revocación

1. El administrador de la zona coloca la llave pública en un registro TXT con el nombre `_dpkirzk` y como cadena coloca la llave pública en codificación Base58Check.

2. Se genera y transmite a la red una transacción `OP_QRTPK` que emplea como dirección la dirección predefinida para avisos de revocación.

3. Cada nodo que recibe la transacción verifica que esté bien formada y coteja contra el servidor DNS la llave pública. En caso de coincidir se agrega la cola para ser minada y a la lista temporal de llaves no confiables, finalmente se transmite a los demás nodos. En caso contrario es descartada.

4. Un nodo minero la agrega a un nuevo bloque.

5. Se debe esperar una cantidad de bloques c para que la llave sea tomada como revocada, mientras tanto se encuentra en el estado de no confiable.

En la figura 7.2 se muestra un diagrama del proceso.

Renovación

La operación de renovación es prácticamente igual a la de alta. La única diferencia es que se indica que se está sustituyendo a una llave ya existente. El motivo de esta diferencia es que se considera la posibilidad de que con la llave privada de la zona se firmen certificados de varios equipos en la zona así como la emisión de certificados para firma de correo electrónico a usuarios que tienen cuentas de correo que emplean la zona. Debido a esto se contempla un periodo de transición entre las llaves públicas de la zona para permitir una migración transparente.

Por ejemplo, supongamos que usted es el administrador de la zona `cs.cinvestav.mx` en este caso usted después de haber agregado la llave pública al blockchain firmó 20 certificados para servidores y 60 certificados para firma de correo electrónico de sus usuarios. En el caso de que no existiera el periodo de transición usted debería cada y años registar una nueva llave pública antes de que su anterior llave sea dada de baja y emitir lo más rápidamente posible los nuevos certificados para sus servidores y usuarios. Durante el periodo que usted realiza esta acción, quién visite alguno de los sitios que usted administra recibirá una alerta de que la llave pública no es válida, así como los mensajes de correo electrónico firmados por sus usuarios serían reportados como no confiables. Como esto puede crear pérdida de confianza en el contenido proveniente de la zona `cs.cinvestav.mx` sería mejor tener una opción en la cual se pueda hacer una transición paulatina y sin que los contenidos de `cs.cinvestav.mx` sean vistos como no confiables.

En la figura 7.3 se muestra una línea del tiempo para la renovación de una llave.

7.3.5. Generación de certificados

Ya que se ha publicado la llave pública de la zona en el blockchain y esta se encuentre confirmada, se puede proceder a generar los certificados correspondientes.

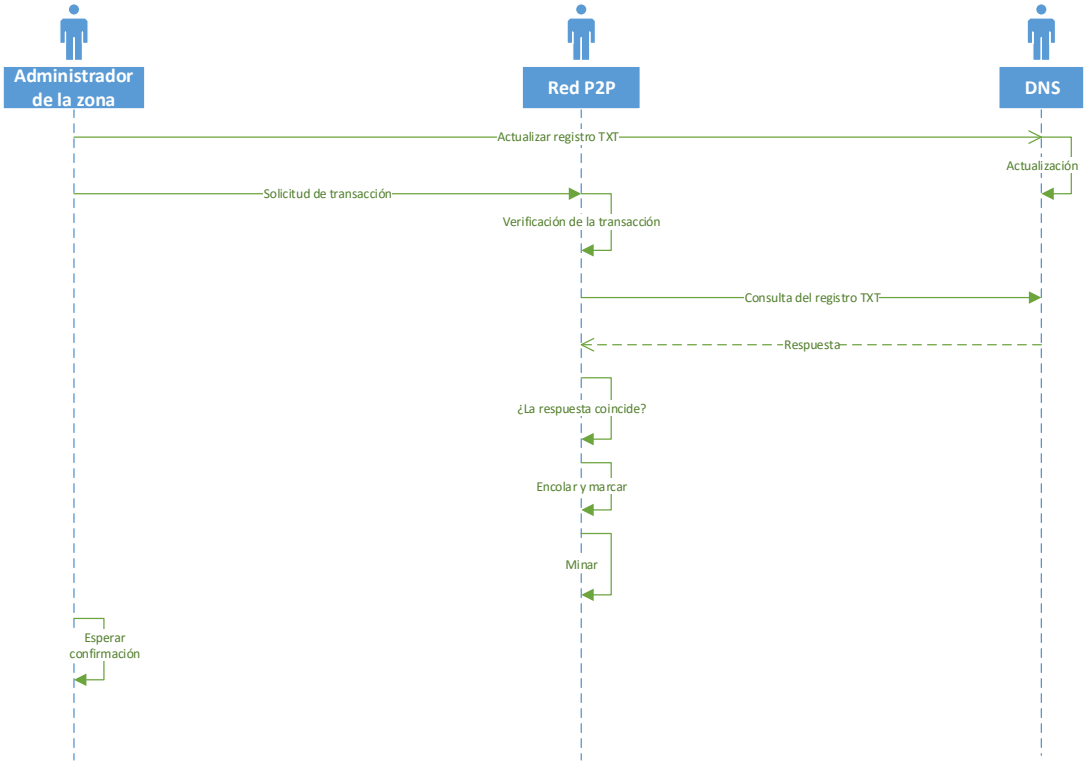


Figura 7.2: Diagrama del proceso de revocación de una llave pública

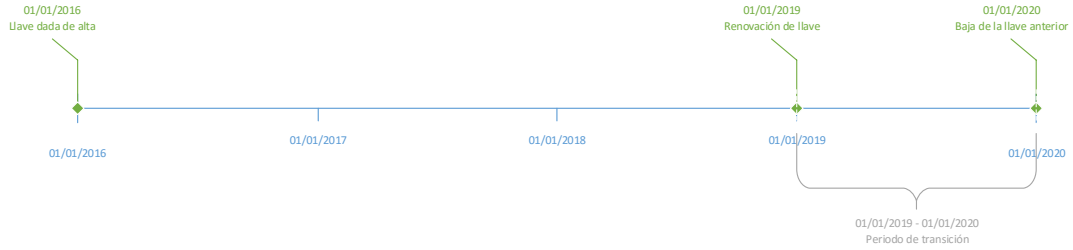


Figura 7.3: Línea del tipo para la renovación de una llave existente

El primer certificado que debe de ser emitido es un certificado autofirmado al cual en el campo de `CertificateIssuerUniqueId` se le coloca el TXID de donde se dio de alta la llave empleada para el certificado autofirmado. También debe de colocarse el campo `nameConstraints` especificando la zona.

Ya que se tiene el certificado autofirmado se pueden emitir los certificados de servidores y si es el caso los de los usuarios para su uso en firma de correo electrónico. Todos estos deben de incluir en su cadena de certificación el certificado autofirmado de la zona.

7.3.6. Construcción del camino de certificación

La forma en que se construye el camino de certificación es la siguiente:

1. Un cliente recibe un certificado al conectarse a un servicio web o al recibir un correo. Éste debe de contener toda la cadena de certificados necesaria para su verificación.
2. El cliente verifica cada uno de los certificados, si todos los datos son correctos y se llega al certificado autofirmado se consulta la lista de llaves no confiables así como de llaves revocadas. Si la llave no se encuentra en ninguna de estas listas se consulta la transacción con el TXID indicado, si la información de la llave y la zona concuerdan se completa el camino de certificación.

7.3.7. Especificación

En las siguientes subsecciones se describen los detalles del diseño propuesto.

Prueba de trabajo

La prueba de trabajo empleada es similar a la de Bitcoin, la diferencia es que se emplea el algoritmo SHA-384. El motivo de esta decisión surge a partir de varios estudios como los presentados en [Beikverdi and Song, 2015] y [Gervais et al., 2014], donde se presenta la preocupante situación de que el grueso del poder de cómputo empleado en Bitcoin, pero sobre todo en el minado unificado, está empezando a ser concentrado por unas pocas entidades los cual es un gran peligro para la operación descentralizada.

Al no existir hardware especializado para este algoritmo se puede contar con una base más democrática. Además al no ser este sistema pensado como una criptomoneda, lo cual incentiva al uso máximo de recursos por parte del usuario para obtener la mayor ganancia posible, se puede realizar una implementación que realice un minado constante que no consuma mínimos recursos y por esto pueda ser una opción viable para instalar en una amplia base de equipos cuyos usuarios estén interesados en construir y colaborar con un Internet más seguro.

Curva elíptica

Se emplea la curva estandarizada `secp384r1`. Esta posee un nivel de seguridad más alto que el proporcionado por la curva empleada en Bitcoin, además de que se apega a las recomendaciones emitidas por NIST para usos más allá del año 2030.

Tamaño de bloque

El tamaño máximo del bloque es de 3 MB.

Tiempo entre bloques

Se implementa un tiempo aproximado de 1 hora entre bloques, esto debido a que el tipo de servicio no requiere de una rápida confirmación de las transacciones.

Ajuste de dificultad

El ajuste de la dificultad se realiza cada 336 bloques, teniendo un tiempo esperado de 14 días entre cada ajuste.

Divisibilidad

Se definió una divisibilidad de hasta 1/100. Esto debido que los usuarios están acostumbrados a manejar en la vida diaria este tipo de subunidades.

Periodo de vigencia

Cada llave será válida por 26280 bloques lo cual se estima en 3 años. Adicionalmente si se indica al momento de renovar la llave que se va a emplear el tiempo de transición se agregan 8760 bloques, que se estima se generen en un año.

Purga de certificados vencidos

El contenido del blockchain se purga cada 34176 bloques, sólo los encabezados desde el inicio del blockchain se conservan. Para evitar que queden fichas huérfanas es necesario emitir una transacción `OP_QSLTB` antes de que la transacción de acuñado o una tranascción `OP_QSLTB` previa sea eliminada, lo cual ocasionará que las fichas restantes se pierdan.

Transacción

Tamaño	Campo	Descripción
1 byte	Versión	Reglas a las cuales se apega la transacción
1-9 bytes	Total de entradas	El número de entradas que se incluyen
Variable	Entradas	Una o más entradas de la transacción
1-9 bytes	Total de salidas	El número de salidas que se incluyen
Variable	Salidas	Una o más salidas de la transacción

Encabezado del bloque

Bytes	Nombre	Tipo de dato	Descripción
1	Versión	uint8_t	La versión del bloque empleado, esto determina las reglas de validación que se deben de seguir

Bytes	Nombre	Tipo de dato	Descripción
48	Picadillo del bloque previo	char[48]	Un picadillo SHA-384(SHA-384()) del encabezado del bloque previo. Su función es que ningún bloque se pueda modificar sin tener que modificar todos los bloques posteriores.
48	Raíz Merkle	char[48]	Un picadillo SHA-384(SHA-384()). La raíz Merkle se deriva de los picadillos de todas las transacciones incluidas en el bloque, asegurando de esta manera que ninguna transacción sea modificada sin tener que modificar el encabezado
8	Fecha	uint64_t	La fecha, en formato UNIX, cuando el bloque comenzó a ser minado (de acuerdo al minero). Debe ser mayor o igual a la mediana de los 11 bloques previos. Los nodos completos no aceptan encabezados con más de dos horas en el futuro, de acuerdo a su reloj.
8	Objetivo	uint64_t	El valor objetivo el cual el picadillo de este bloque debe ser menor o igual.
8	Número arbitrario	uint64_t	Un número arbitrario el cual los mineros modifican para poder producir un picadillo con valor menor o igual al objetivo.

Entrada

Tamaño	Campo	Descripción
48 bytes	Picadillo de la transacción	Apuntador a la transacción que contiene la UTXO a gastar
4 bytes	Índice de la salida	El índice de la UTXO a ser gastada. Inicia en 0.
1-9 bytes	Tamaño del script de desbloqueo	Tamaño en bytes que tiene el script de desbloqueo
Variable	Script de desbloqueo	Un script el cual cumple las condiciones del script de bloqueo de la UTXO

Entrada de acuñado

Tamaño	Campo	Descripción
48 bytes	Picadillo de la transacción	Todos los bits en 0
4 bytes	Índice de la salida	Todos los bits en 1
1-9 bytes	Tamaño de la información de acuñado	Tamaño en bytes que tiene la información de acuñado. 2 a 100 bytes.
Variable	Información de acuñado	Información arbitraria empleada para números aleatorios adicionales e información de minado. Debe iniciar con la altura del bloque.

Salida

Tamaño	Campo	Descripción
8 bytes	Monto	Cantidad de fichas
1-9 bytes	Tamaño del script de bloqueo	Tamaño en bytes que tiene el script de bloqueo
Variable	Script de bloqueo	Un script el cual define las condiciones que se deben de cumplir para poder gastar el monto

Transacción OP_ADDPK

La forma en que se compone una transacción de tipo `OP_ADDPK` es la siguente:

Tamaño	Campo	Descripción
1 byte	Tamaño	La cantidad de caracteres que posee el nombre de la zona
Variable	Nombre	El nombre de la zona que registra la llave pública
1 byte	Servicios de estado	Cantidad de servidores de verificación de estado (0-5)
Variable	Lista de servidores de estado	El tipo y la dirección donde se consulta el servicio de verificación de estado.
48 bytes	Llave a sustituir	En el caso de que se esté renovando una llave y se desee usar el periodo de transición se indica el TXID de la llave a renovar, en caso contrario se colocan todos los bits en 0.

A continuación se muestra un ejemplo en formato JSON:

```
{
    "zone": "google.com",
    "status": {
        "0":[1,"http://pki.google.com/GIAG2.crt"],
        "1":[2, "http://clients1.google.com/ocsp"]
    }
    "succeed":"cb3e7d6ac1141cd9ced135ddc2752f0f27fd60dd
    6b644b39f993b3153b1bfb5364defaab81f13f9cac117bee3670216b"
}
```

Transacción OP_QRTPK

La transacción OP_QRTPK que indica la revocación de una llave, contiene la siguiente información:

Tamaño	Campo	Descripción
48 bytes	TXID	El TXID de la llave que se desea revocar

Se realizará una verificación mediante DNS, es necesario colocar la llave pública a revocar en un registro TXT con el nombre `_dpkirzk`.

Transacción OP_QSLTB

La transacción `OP_QSLTB` genera una única salida la cual agrega todas sus entradas. Este tipo de transacción indica que estas fichas fueron creadas en un tiempo previo pero que pronto las operaciones que las generaron están por ser eliminadas. Para que las entradas sean válidas deben de provenir de una transacción de acuñado o de una transacción `OP_QSLTB` previa que siga existiendo en el blockchain al momento de minar el bloque.

7.3.8. Posibles ataques

Los posibles ataques que puede sufrir el sistema son derivados de diseño inspirado en Bitcoin, además de envenenamiento de *cache* de DNS.

50 % + 1

El que un grupo obtenga más del 50 % del poder de cómputo de todo el sistema, al igual que en Bitcoin, le permite poder tener el control de los nuevos bloques.

Privacidad

En el caso del sistema propuesto, la privacidad del sitio visitado no se toma como un objetivo, sino la privacidad de los datos transmitidos con éste. Aunque el sistema no contempla este objetivo, en el caso de así hacerlo no sería de gran ayuda ya que todos los datos del sistema de DNS viajan en claro. Siendo aun posible el conocer los sitios visitados. Para proveer privacidad en este escenario es apto el uso de Tor o de servicios de VPN.

Denegación de servicio

Se corre el riesgo de que algunos nodos maliciosos no respondan a las solicitudes sobre alguna llave en especifico, haciendo creer al cliente que este sitio no ha dado de alta su llave en el blockchain y a la vez que desconfíe del certificado provisto por el servidor web. Para evitar esto se recomienda tener conexión con varios nodos, los cuales se encuentren distribuidos en diferentes regiones. Así como, en caso de ser posible, tener un nodo completo de forma local para no depender de consultas externas.

Envenenamiento de cache *de DNS*

Un adversario podría intentar evitar que una llave llegue al blockchain envenenando el *cache* de varios servidores DNS, esto haría que los nodos que consulten estos servidores desechen la transaccionan de alta o renovación debido a que las llaves públicas no coinciden al ser verificadas contra el DNS. Para poder lograr que esto sea exitoso requeriría que más de la mitad de los nodos no puedan verificar la llave pública. Debido a que se plantea que el protocolo sea usado en todo el mundo, lograr este ataque es improbable ya que se debería envenenar demasiados servidores de DNS lo cual sería detectado.

7.3.9. Ambiente de prueba y desarrollo

Para poder realizar el desarrollo y la verificación del diseño propuesto se realizó la construcción de un ambiente el cual consta de los siguientes elementos:

- Servidor DNS (BIND 9.9.7)
- Servidor PKI (EJBCA 6.2.0)
- Servidor de versionamiento (Perforce Helix 2015.1.1041035)
- Servidor con Bitcoin Core (0.10.2)
- Servidor de desarrollo (OpenSuSE 13.2)

Todos estos son máquinas virtuales dentro de un servidor VMware vSphere 6 con las siguientes características:

- Procesador Intel i7 2600K @ 3.40 GHz
- 24 Gb de memoria RAM
- 120 GB de almacenamiento SSD
- 500 GB de almacenamiento HDD
- 3 NICs de 1 Gbps

Servidor DNS

El servidor de nombres de dominio se implemento con la finalidad de verificar la factibilidad del uso de registros TXT para el almacenamiento de las llaves públicas de la zona. Esto se realizó generando la zona `crypto.cs.cinvestav.mx` para la cual el servidor posee autoridad sobre esta.

Se realizaron las siguientes pruebas:

- Funcionalidad de los servicios de DNS.
- Resolución de nombres de la zona.
- Inclusión de un registro TXT.
- Consulta desde un cliente de registros TXT (ver figura 7.4).

Servidor PKI

Para verificar que la creación y manejo de certificados no fuera un impedimento para el uso del sistema se instaló un servidor con el software EJBA, el cual es una PKI de código abierto que provee todas las funcionalidades necesarias para la gestión de un certificado a lo largo de su ciclo de vida.

Se verificaron las siguientes operaciones:

- Alta de una CA con una llave privada pregenerada.
- Alta de una CA con una llave privada nueva.
- Generación de un certificado autofirmado para la CA.

```
abraham@sierra:~> dig TXT  _dpkinpk.crypto.cs.cinvestav.mx

; <<>> DiG 9.9.6-P1 <<>> TXT _dpkinpk.crypto.cs.cinvestav.mx
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56379
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; QUESTION SECTION:
;_dpkinpk.crypto.cs.cinvestav.mx. IN    TXT

;; ANSWER SECTION:
_dpkinpk.crypto.cs.cinvestav.mx. 10800 IN TXT   "1QBZwKZFWFKnYodZdjQ9ufa
3eWhS55hgh54hbZ2Vx4i48wBd7Fc1h"

;; AUTHORITY SECTION:
crypto.cs.cinvestav.mx. 10800   IN      NS      aleph.crypto.cs.cinvestav.mx.

;; ADDITIONAL SECTION:
aleph.crypto.cs.cinvestav.mx. 10800 IN  A       172.20.10.2

;; Query time: 3 msec
;; SERVER: 172.20.10.1#53(172.20.10.1)
;; WHEN: Tue Dec 10 19:36:56 CST 2015
;; MSG SIZE  rcvd: 151
```

Figura 7.4: Consulta de un registro TXT

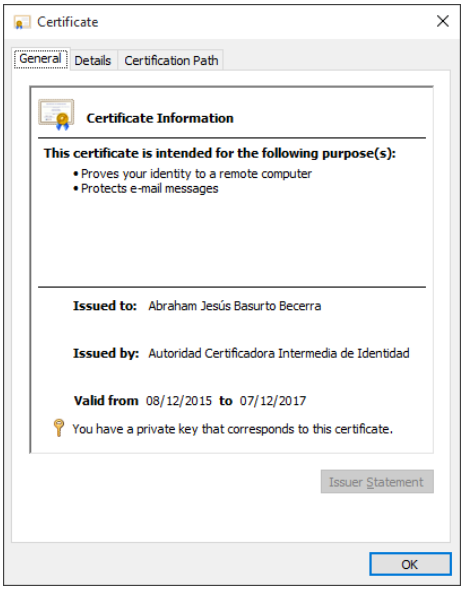


Figura 7.5: Ejemplo de un certificado emitido

- Emisión de certificados firmados por la CA.
- Generación de sub-CAs.
- Emisión de certificados firmados por una sub-CA (ver figura 7.5).
- Emisión de listas de revocación.
- Funcionalidad de OCSP.
- Consulta de certificados emitidos.

Servidor de versionamiento

El servidor de versionamiento se instaló con la finalidad de llevar un control sobre los cambios realizados tanto al código de Bitcoin Core, así como al documento de tesis. Este servidor no es indispensable para el funcionamiento del sistema como tal, pero provee de una valiosa ayuda al momento de realizar cualquier desarrollo.

Servidor de Bitcoin Core

Este servidor sirvió un doble propósito, primero el servir como plataforma de pruebas para poder comprender los conceptos del uso de Bitcoin y de su ecosistema; y segundo, el poder procesar los pagos que empleaban bitcoins del sistema implementado para Latincrypt 2015.

7.3.10. Metodología de prueba

Para verificar el funcionamiento del diseño se propone crear una máquina virtual con los componentes de sistema operativo y software mínimos necesarios para poder ejecutar el cliente creado. Ya creada, dicha imagen se puede emplear para generar fácilmente una red con múltiples clientes interactuando entre ellos. Inicialmente estarían confinados al ambiente de prueba empleando el servidor DNS local para realizar la verificación de llaves públicas y posteriormente se emplearían nodos adicionales fuera del ambiente de prueba para observar el funcionamiento del sistema a través de Internet.

Para el ambiente de prueba el tiempo entre bloque y bloque sería reducido a periodos cortos entre uno y cinco minutos para poder realizar la verificación de manera más fluida. Posteriormente se podría pasar al uso de los tiempos definitivos y verificar el correcto funcionamiento del sistema.

CAPÍTULO 8

CONCLUSIONES Y TRABAJO FUTURO

If we knew what it was we were doing, it would not be called research, would it?

—ALBERT EINSTEIN

Dado que cada vez se ha hecho más popular el uso de la criptomoneda Bitcoin, es necesario conocer los fundamentos en que se basa así como tener una comprensión del protocolo que emplea y sus posibles vulnerabilidades.

Bitcoin no posee un una formalización de su modelo de seguridad, ha habido varias propuestas pero tienen un enfoque limitado. Es importante el investigar en este tema ya que hasta el momento no se puede probar de manera convincente si el modelo de Bitcoin es seguro o no.

Dentro de la literatura existen muy pocas obras que realicen una descripción de las ideas que Bitcoin incorpora así como de su operación, y de las pocas que existen la mayoría lo abordan con un tono sensacionalista y especulativo sobre cómo éste podría remplazar al dinero fiduciario o brindar grandes ganancias monetarias. Aparte de los libros en este tono, existen otros que son realmente guías de cómo usar Bitcoin a nivel de usuario. Finalmente están los que describen su funcionamiento, estos pueden ser contados con los dedos de una mano y posiblemente nos sobren dedos. Cabe destacar que a lo largo de la realización de este trabajo no se logró encontrar un libro o publicación en español que realizara una descripción tan a detalle

del funcionamiento y conceptos de Bitcoin como se ha realizado en este trabajo.

De este trabajo se puede tener una amplia visión de como la criptografía es empleada en el mundo real, desde cuando navegamos en Internet hasta cuando realizamos pagos a través de una criptomoneda. Además de las implicaciones prácticas y problemas que surgen, como lo es la distribución y verificación de autenticidad de las llaves.

También se puede apreciar el gran problema que existe con la actual implementación de la infraestructura de llave pública y lo necesario que es el encontrar alternativas viables, para de esta manera poder asegurar la confidencialidad e integridad en nuestras comunicaciones.

Como conclusión final se puede indicar que la aportación más importante que ha brindado Bitcoin no es como tal la criptomoneda, sino la idea del blockchain la cual apenas empieza a permear en otros sectores diferentes al de la ciencia, un ejemplo es el banco español Santander el cual ha buscado usos potenciales para esta tecnología y en un reporte [Belinky et al., 2015] indica que tiene el potencial de poder reducir costos de operación en hasta 20 mil millones de dólares al año para los bancos en las operaciones de pagos internacionales, bonos y cumplimento regulatorio.

8.1. Trabajo futuro

Como todo sistema crítico tanto los requerimientos del sistema así como el diseño se expresan con detalle y son analizados cuidadosamente antes de que comience la implementación. En la realización de este trabajo de tesis se ha realizado una amplia labor de investigación y de diseño. Es de gran importancia tener la retroalimentación que proporciona la implementación y realización de pruebas ya que la implementación correcta de primitivas y protocolos criptográficos es en sí toda un área de estudio. Debido a la limitante del tiempo no ha sido posible realizar una implementación completa del diseño por lo que es el principal trabajo pendiente a realizar.

Este trabajo de tesis sólo se enfocó en la utilización de la alternativa de infraestructura de llave pública propuesta para su uso con el protocolo HTTPS, sería interesante el analizar su factibilidad de uso en otras aplicaciones como lo son notarización de documentos digitales y firmado de software.

APÉNDICE A

REFERENCIA DE SCRIPT

A continuación se presenta una lista de todas las instrucciones de *Script*.

Existen instrucciones las cuales existieron en las versiones iniciales de Bitcoin pero fueron removidas debido a la posibilidad de que existiera un eror de implementación en el cliente. Esto fue derivado del error encontrado en la instrucción `OP_LSHIFT` el cual podía hacer que cualquier cliente de Bitcoin fallara, adicionalmente existieron errores los cuales permitían gastar los bitcoins de cualquier otra persona. Las instrucciones que fueron removidas frecuentemente son denominadas como deshabilitadas, aunque esto es meramente terminología ya que las instrucciones no existen más en el protocolo de Bitcoin y no existen planes para reincorporar alguna de estas. Las instrucciones removidas son incluidas en la lista debido a su interés histórico.

Con el propósito de que nuevas instrucciones puedan ser agregadas al protocolo en el futuro se incluyeron las instrucciones `OP_NOP1` - `OP_NOP10`.

El valor de *falso* es cero, cero negativo o un arreglo vació; cualquier otro valor es *verdadero*.

A.1. Constantes

Normalmente para meter constantes a la pila en los *scripts* se emplea el código en vez de la instrucción.

Instrucción	Código	Entrada	Salida	Descripción
`OP_0`, `OP_FALSE`	0x00	nada	(vacio)	Un arreglo vacio se mete a la pila. (No confundir con `OP_NOP`)
	0x01 - 0x4b	(especial)	datos	Los siguientes N bytes se meten a la pila, donde N = [1,75].
`OP_PUSHDATA1`	0x4c	(especial)	datos	El siguiente byte indica la cantidad de bytes a meter a la pila.
`OP_PUSHDATA2`	0x4d	(especial)	datos	Los siguientes dos bytes indican la cantidad de bytes a meter a la pila.
`OP_PUSHDATA4`	0x4e	(especial)	datos	Los siguientes cuatro bytes indican la cantidad de bytes a meter a la pila.
`OP_1NEGATE`	0x4f	nada	-1	El número -1 se mete a la pila.
`OP_1`, `OP_TRUE`	0x51	nada	1	El número 1 se mete a la pila.
`OP_2` - `OP_16`	0x52 - 0x60	nada	2-16	El número indicado (2-6) se mete a la pila.

A.2. Control de flujo

Instrucción	Código	Entrada	Salida	Descripción
`OP_NOP`	0x61	nada	nada	Instrucción ociosa, no hace nada.
`OP_IF`	0x63	<expresión> if [sentencias] [else [sentencias]]* endif		Si el valor del tope de la pila es diferente de 0, las sentencias son ejecutadas. El valor del tope de la pila se elimina.

Instrucción	Código	Entrada	Salida	Descripción
OP_NOTIF	0x64	<expresión> if [sentencias] [else [sentencias]]* endif		Si el valor del tope de la pila es 0, las sentencias son ejecutadas. El valor del tope de la pila se elimina.
OP_ELSE	0x67	<expresión> if [sentencias] [else [sentencias]]* endif		Si las sentencias de instrucciones OP_IF, OP_NOTIF o OP_ELSE precedentes no fueron ejecutadas, estas sentencias se ejecutan. En el caso contrario no se ejecutan.
OP_ENDIF	0x68	<expresión> if [sentencias] [else [sentencias]]* endif		Finaliza un bloque if/else. Todos los bloques deben de ser terminados o la tranascción será inválida. Un OP_ENDIF sin un OP_IF previo también es inválido..
OP_VERIFY	0x69	verdadero/ falso	nada /error	Marca la transacción como invalida si el valor del tope de la pila no es verdadero.
OP_RETURN	0x6a	nada	error	Marca la transacción como invalida. Una forma estándar de agregar información adicional a las transacciones es agregar una salida con valor 0 con un *script* de bloqueo que consiste de una instrucción OP_RETURN seguida de una operación PUSHDATA. Este tipo de salidas son incobrables.

A.3. Manejo de pila

Instrucción	Código	Entrada	Salida	Descripción
OP_TOALTSTACK	0x6b	x_1	(alt)x_1	Coloca la entrada en el tope de la pila alterna. La remueve de la pila principal.
OP_FROMALTSTACK	0x6c	(alt)x_1	x_1	Coloca la entrada en el tope de la pila principal. La remueve de la pila alterna.
OP_IFDUP	0x73	x	x / x x	Si el valor del tope de la pila es diferente de cero se duplica.
OP_DEPTH	0x74	nada	<tamaño de la pila>	Coloca el número indicado de elementos en la pila.
OP_DROP	0x75	x	nada	Remueve el tope de la pila.
OP_DUP	0x76	x	x x	Duplica el tope de la pila.
OP_NIP	0x77	x_1 x_2	x_2	Remueve el segundo elemento de la pila.
OP_OVER	0x78	x_1 x_2	x_1 x_2 x_1	Copia el segundo elemento de la pila y lo coloca en el tope.

Instrucción	Código	Entrada	Salida	Descripción
OP_PICK	0x79	$x_n \ldots x_2 x_1 x_0$	$x_n \ldots x_2 x_1 x_0 x_n$	Copia el n esimo elemento de la pila y lo coloca en el tope.
OP_ROLL	0x7a	$x_n \ldots x_2 x_1 x_0$	$\ldots x_2\ x_1\ x_0\ x_n$	Mueve el n esimo elemento de la pila al tope.
OP_ROT	0x7b	$x_1\ x_2\ x_3$	$x_2\ x_3\ x_1$	Los tres primeros elementos de la pila son rotados a la izquierda.
OP_SWAP	0x7c	$x_1\ x_2$	$x_2\ x_1$	Los dos primeros elementos de la pila son intercambiados.
OP_TUCK	0x7d	$x_1\ x_2$	$x_2\ x_1\ x_2$	El tope de la pila es copiado e insertado despúes del segundo elemento de la pila.
OP_2DROP	0x6d	$x_1\ x_2$	nada	Remueve los dos primeros elementos de la pila.
OP_2DUP	0x6e	$x_1\ x_2$	$x_1\ x_2\ x_1\ x_2$	Duplica los dos primeros elementos de la pila.
OP_3DUP	0x6f	$x_1\ x_2\ x_3$	$x_1 x_2 x_3 x_1 x_2 x_3$	Duplica los tres primeros elementos de la pila.
OP_2OVER	0x70	$x_1\ x_2\ x_3\ x_4$	$x_1 x_2 x_3 x_4 x_1 x_2$	Copia el segundo par de elementos desde el tope de la pila al tope.
OP_2ROT	0x71	$x_1 x_2 x_3 x_4 x_5 x_6$	$x_3 x_4 x_5 x_6 x_1 x_2$	Los elementos 5 y 6 son movidos al tope de la pila.
OP_2SWAP	0x72	$x_1\ x_2\ x_3\ x_4$	$x_3\ x_4\ x_1\ x_2$	Los dos primeros pares de elementos de la pila son intercambiados.

A.4. Cadenas

El *script* es abortado y se indica error en el caso de que alguna de las instrucciones deshabilitadas se encuentre en el *script*.

Instrucción	Código	Entrada	Salida	Descripción
OP_CAT	0x7e	$x_1\ x_2$	salida	Concatena dos cadenas. **Deshabilitada**.
OP_SUBSTR	0x7f	entrada inicio tamaño	salida	Regresa una sección de la cadena. **Deshabilitada**.
OP_LEFT	0x80	entrada tamaño	salida	Conserva únicamente los caracteres a la izquierda del punto especificado. **Deshabilitada**.
OP_RIGHT	0x81	entrada tamaño	salida	Conserva únicamente los caracteres a la derecha del punto especificado. **Deshabilitada**.

OP_SIZE	0x82	entrada	entrada tamaño	Calcula el tamaño de la cadena en el tope de la pila e inserta este valor a la pila. La cadena no se saca de la pila.

A.5. Operaciones lógicas

El *script* es abortado y se indica error en el caso de que alguna de las instrucciones deshabilitadas se encuentre en el *script*.

Instrucción	Código	Entrada	Salida	Descripción
OP_INVERT	0x83	x_1	$\neg x_1$	Invierte todos los bits de la entrada. **Deshabilitada**.
OP_AND	0x84	x_1 x_2	$x_1 \wedge x_2$	Operación AND a nivel de bits en los dos elementos superiores de la pila. **Deshabilitada**.
OP_OR	0x85	x_1 x_2	$x_1 \vee x_2$	Operación OR a nivel de bits en los dos elementos superiores de la pila. **Deshabilitada**.
OP_XOR	0x86	x_1 x_2	$x_1 \oplus x_2$	Operación XOR a nivel de bits en los dos elementos superiores de la pila. **Deshabilitada**.
OP_EQUAL	0x87	x_1 x_2	verdadero / falso	Saca los dos elementos superiores de la pila, si son iguales inserta 1 y en el caso contrario inserta 0.
OP_EQUALVERIFY	0x88	x_1 x_2	nada / error	Igual a OP_EQUAL, pero ejecuta OP_VERIFY posteriormente.

A.6. Operaciones aritméticas

La entradas a operaciones aritméticas están limitadas a enteros signados de 32 bits, las salidas pueden desbordarse.

En caso de proporcionar una entrada con un valor mayor a 4 bytes el *script* es abortado y se indica error. También el *script* es abortado y se indica error en el caso de que alguna de las instrucciones deshabilitadas se encuentre en el *script*.

Instrucción	Código	Entrada	Salida	Descripción
OP_1ADD	0x8b	x	$x + 1$	Se le suma 1 a la entrada.
OP_1SUB	0x8c	x	$x - 1$	Se le resta 1 a la entrada.

Instrucción	Código	Entrada	Salida	Descripción
OP_2MUL	0x8d	x	$x \times 2$	La entrada se multiplica por 2. **Deshabilitada**.
OP_2DIV	0x8e	x	$x \div 2$	La entrada se divide entre 2. **Deshabilitada**.
OP_NEGATE	0x8f	x	$-(x)$	Se invierte el signo de la entrada.
OP_ABS	0x90	x	$\|x\|$	Se toma el valor absoluto de la entrada.
OP_NOT	0x91	entrada	0 / 1	Si la entrada es 0 o 1 se invierte. En cualquier otro caso la salida es 0.
OP_0NOTEQUAL	0x92	entrada	0 / 1	Si la entrada es 0 la salida es 0. En cualquier otro caso la salida es 1.
OP_ADD	0x93	a b	$a + b$	A a se le suma b.
OP_SUB	0x94	a b	$a - b$	A a se le resta b.
OP_MUL	0x95	a b	$a \times b$	a multiplicado por b. **Deshabilitada**.
OP_DIV	0x96	a b	$a \div b$	a dividido entre b. **Deshabilitada**.
OP_MOD	0x97	a b	a	Se regresa el residuo de la división de a entre b. **Deshabilitada**.
OP_LSHIFT	0x98	a b	$a << b$	Se le realizan b corrimientos a la izquierda a a, conservando el signo. **Deshabilitada**.
OP_RSHIFT	0x99	a b	$a >> b$	Se le realizan b corrimientos a la derecha a a,conservando el signo. **Deshabilitada**..
OP_BOOLAND	0x9a	a b	0 / 1	Si a y b son diferentes de 0 la salida es 1. En cualquier otro caso la salida es 0.
OP_BOOLOR	0x9b	a b	0 / 1	Si a o b son diferentes de 0 la salida es 1.En cualquier otro caso la salida es 0.
OP_NUMEQUAL	0x9c	a b	0 / 1	Si los números a y b son iguales la salida es 1. De no ser así la salida es 0.

Instrucción	Código	Entrada	Salida	Descripción
OP_NUMEQUALVERIFY	0x9d	a b	nada / error	Igual que OP_NUMEQUAL, pero OP_VERIFY se ejecuta posteriormente.
OP_NUMNOTEQUAL	0x9e	a b	0 / 1	Si los números a y b son diferentes la salida es 1. De no ser así la salida es 0.
OP_LESSTHAN	0x9f	a b	0 / 1	Si a es menor que b la salida es 1. De no ser así la salida es 0.
OP_GREATERTHAN	0xa0	a b	0 / 1	Si a es mayor que b la salida es 1. De no ser así la salida es 0.
OP_LESSTHANOREQUAL	0xa1	a b	0 / 1	Si a es menor o igual que b la salida es 1. De no ser así la salida es 0.
OP_GREATERTHANOREQUAL	0xa2	a b	0 / 1	Si a es mayor o igual que b la salida es 1. De no ser así la salida es 0.
OP_MIN	0xa3	a b	a / b	Se regresa la entrada con menor valor.
OP_MAX	0xa4	a b	a / b	Se regresa la entrada con mayor valor.
OP_WITHIN	0xa5	x min max	0 / 1	Si x se encuentra en el intervalo especificado (cerrado por la izquierda) la salida es 1.En cualquier otro caso la salida es 0..

A.7. Operaciones criptográficas

Instrucción	Código	Entrada	Salida	Descripción
OP_RIPEMD160	0xa6	entrada	picadillo	Se calcula el picadillo de la entrada usando RIPEMD-160.
OP_SHA1	0xa7	entrada	picadillo	Se calcula el picadillo de la entrada usando SHA-1.
OP_SHA256	0xa8	entrada	picadillo	Se calcula el picadillo de la entrada usando SHA-256.
OP_HASH160	0xa9+	entrada	picadillo	Se calcula el picadillo de la entrada dos veces : primero usando SHA-256 y posteriormente con RIPEMD-160. RIPEMD-160(SHA-256())

Instrucción	Código	Entrada	Salida	Descripción
OP_HASH256	0xaa	entrada	picadillo	Se calcula el picadillo de la entrada dos veces usando SHA-256. SHA-256(SHA-256())
OP_CODESEPARATOR	0xab	nada	nada	Todas las instrucciones de firma verifican las firmas de los datos que se encuentran después de la instrucción OP_CODESEPARATOR más reciente.
OP_CHECKSIG	0xac	firma llave_pública	verdadero /falso	Se calcula en picadillo de todas las entradas, salidas y el script de una transacción (a partír de la instrucción OP_CODESEPARATOR más reciente) . La firma proporcionada a OP_CHECKSIG debe ser una firma válida del picadillo al verificarla con la llave pública proporcionada. Si la firma verifica la salida es 1, en cualquier otro caso la salida es 0.
OP_CHECKSIGVERIFY	0xad	firma llave_pública	nada/falso	Igual que OP_CHECKSIG pero OP_VERIFY se ejecuta posteriormente.

Instrucción	Código	Entrada	Salida	Descripción
OP_CHECKMULTISIG	0xae	$x\ firma_1\ firma_2 \ldots$ <número de firmas> $K^1_{pub}\ K^2_{pub}\ \ldots$ >número de llaves públicas>	verdadero /falso	Verifica la primera firma con cada llave pública hasta que encuentra una que verifique. A partír de la siguiente llave pública, verifica la segunda firma con cada una de las llaves que sobran. El proceso se repite hasta que todas las firmas han verificado o ya no existen llaves públicas que verifiquen. Todas las firmas deben de verificar con alguna de llas llaves públicas provistas. Debido a que las llaves públicas no son usadas de nuevo en caso de no haber verificado, las firmas deben de ser colocadas en el mismo orden que se encuentran en el script de bloqueo. Si todas las firmas son verificadas exitosamente la salida es 1, en otro caso la salida es 0. Debido a un error en la implementación un elemento extra que no es usado se extrae de la pila.
OP_CHECKMULTISIGVERIFY	0xaf	$x\ firma_1\ firma_2 \ldots$ <número de firmas> $K^1_{pub}\ K^2_{pub}\ \ldots$ >número de llaves públicas>	nada/falso	Igual que OP_CHECKMULTISIG pero OP_VERIFY se ejecuta posteriormente.

A.8. Pseudoinstrucciones

Estas instrucciones son usadas de forma interna para ayudar a identificar transacciones. Son inválidas si se emplean en un *script*.

Instrucción	Código	Descripción
OP_PUBKEYHASH	0xfd	Representa el picadillo de una llave pública realizado con OP_HASH160.
OP_PUBKEY	0xfe	Representa una llave pública compatible con OP_CHECKSIG.
OP_INVALIDOPCODE	0xff	Corresponde a cualquier instrucción que no haya sido asignada.

A.9. Instrucciones reservadas

Cualquier código que no es encuentre asignado se considera como reservado. El usar un código sin asignar vuelve a la transacción inválida.

Instrucción	Código	Descripción
OP_RESERVED	0x50	La transacción es inválida a menos que se encuentre en un caso no ejecutado de OP_IF
OP_VER	0x62	La transacción es inválida a menos que se encuentre en un caso no ejecutado de OP_IF
OP_VERIF	0x65	La transacción es inválida aún cuando se encuentre en un caso no ejecutado de OP_IF
OP_VERNOTIF	0x66	La transacción es inválida aún cuando se encuentre en un caso no ejecutado de OP_IF
OP_RESERVED1	0x89	La transacción es inválida a menos que se encuentre en un caso no ejecutado de OP_IF
OP_RESERVED2	0x8a	La transacción es inválida a menos que se encuentre en un caso no ejecutado de OP_IF
OP_NOP1 - OP_NOP10	0xb0 - 0xb9	La instrucción es ignorada. No invalida la transacción.

APÉNDICE B

APLICACIÓN DE PAGO LATINCRYPT 2015

B.1. Esquema de base de datos

Creación de la base de datos.

```
CREATE DATABASE lc2015
    DEFAULT CHARACTER SET utf8
    DEFAULT COLLATE utf8_general_ci;
```

Tabla para los datos del registro de asistentes.

```
CREATE TABLE register (
    RegID INT NOT NULL AUTO_INCREMENT,
    CustID varchar(20) COLLATE utf8_bin NOT NULL,
    FirstName varchar(40) COLLATE utf8_bin NOT NULL,
    LastName varchar(40) COLLATE utf8_bin NOT NULL,
    Affilation varchar(40) COLLATE utf8_bin NOT NULL,
    Email varchar(40) COLLATE utf8_bin NOT NULL,
    Country varchar(50) COLLATE utf8_bin DEFAULT NULL,
    Address varchar(80) COLLATE utf8_bin DEFAULT NULL,
    City varchar(40) COLLATE utf8_bin DEFAULT NULL,
    ZipCode varchar(8) COLLATE utf8_bin DEFAULT NULL,
    Phone varchar(15) COLLATE utf8_bin DEFAULT NULL,
    Ext varchar(5) COLLATE utf8_bin DEFAULT NULL,
    RegOnTime TINYINT DEFAULT '0',
    RegularNum TINYINT DEFAULT '0',
    StudentNum TINYINT DEFAULT '0',
    GuestNum TINYINT DEFAULT '0',
    RegDate DATETIME DEFAULT CURRENT_TIMESTAMP,
    PayMethod TINYINT DEFAULT '0',
    SubTotal INT DEFAULT '0',
    EmailSent TINYINT DEFAULT '0',
    UNIQUE (CustID),
    PRIMARY KEY (RegID)
);
```

Tabla de pagos con tarjeta de crédito.

```
CREATE TABLE ccpay (
    PayID INT NOT NULL AUTO_INCREMENT,
    RegID INT NOT NULL,
    FolioVenta varchar(25) COLLATE utf8_bin DEFAULT NULL,
    HasPayed TINYINT DEFAULT '0',
    PRIMARY KEY (PayID),
    FOREIGN KEY (RegID) REFERENCES register(RegID) ON DELETE CASCADE
);
```

Tabla de pagos con bitcoins.

```
CREATE TABLE btcpay (
```

```
    BtcID INT NOT NULL AUTO_INCREMENT,
    RegID INT NOT NULL,
    Address varchar(40) COLLATE utf8_bin DEFAULT NULL,
    HasPayed TINYINT DEFAULT '0',
    ExchRate varchar(15) COLLATE utf8_bin DEFAULT NULL,
    BtcCost varchar(15) COLLATE utf8_bin DEFAULT NULL,
    PRIMARY KEY (BtcID),
    FOREIGN KEY (RegID) REFERENCES register(RegID) ON DELETE CASCADE
);
```

Tabla con direcciones de pago de reserva en caso de no poder contactar al servidor de Bitcoin.

```
CREATE TABLE btcbcp (
    BcpID INT NOT NULL AUTO_INCREMENT,
    Address varchar(40) COLLATE utf8_bin DEFAULT NULL,
    IsUsed TINYINT DEFAULT '0',
    PRIMARY KEY (BcpID)
);
```

Tabla con los datos de los usuarios administrativos del sistema.

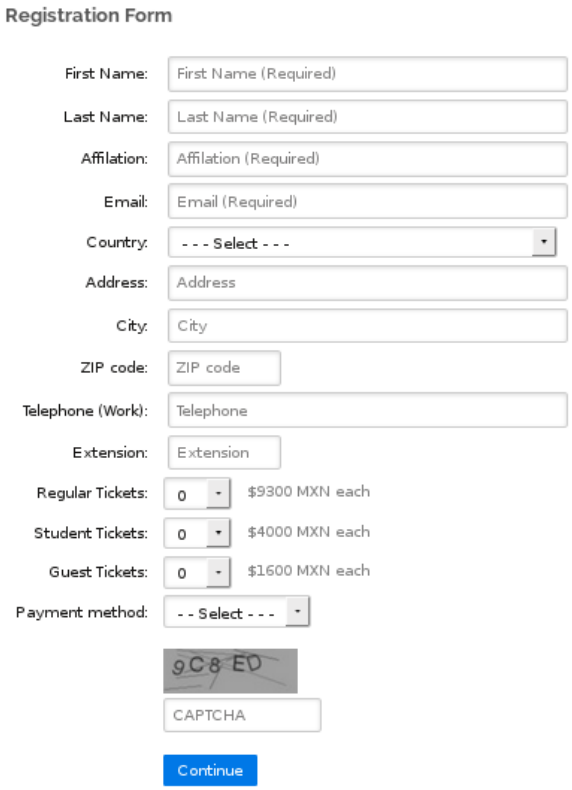
```
CREATE TABLE whois (
    UID INT NOT NULL AUTO_INCREMENT,
    User varchar(15) COLLATE utf8_bin NOT NULL,
    Salt varchar(5) COLLATE utf8_bin NOT NULL,
    Hash varchar(64) COLLATE utf8_bin NOT NULL,
    UNIQUE (User),
    PRIMARY KEY (UID)
);
```

Tabla para registro de becas solicitadas y otorgadas.

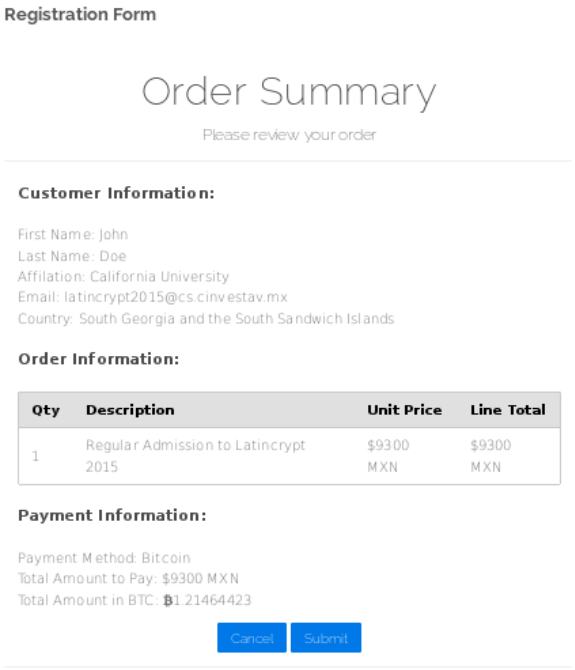
```
CREATE TABLE schreq (
    ReqID INT NOT NULL AUTO_INCREMENT,
    RegID INT NOT NULL,
    IsAnswered TINYINT DEFAULT '0',
    IsGranted TINYINT DEFAULT '0',
    PRIMARY KEY (ReqID),
    FOREIGN KEY (RegID) REFERENCES register(RegID) ON DELETE CASCADE
);
```

B.2. Interfaz

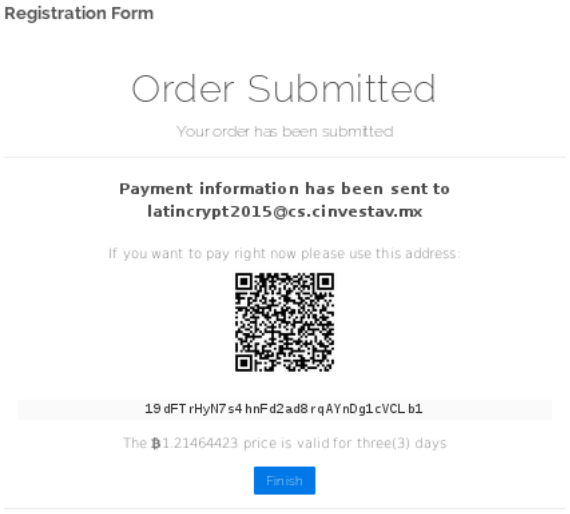
Forma de registro.



Resumen de la orden.



Confirmación de la orden y datos de pago.



REFERENCIAS

[Adams and Lloyd, 2002] Adams, C. and Lloyd, S. (2002). *Understanding PKI: Concepts, Standards, and Deployment Considerations*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, United States of America, 2nd edition.

[Aitchison, 2011] Aitchison, R. (2011). *Pro DNS and BIND 10*. Apress, Berkely, CA, USA, 1st edition.

[Androulaki et al., 2012] Androulaki, E., Karame, G., Roeschlin, M., Scherer, T., and Capkun, S. (2012). Evaluating User Privacy in Bitcoin. Cryptology ePrint Archive, Report 2012/596. `http://eprint.iacr.org/`.

[ANSI, 2005] ANSI (2005). *ANSI X9.62: Public Key Cryptography for the Financial Services Industry - The Elliptic Curve Digital Signature Algorithm (ECDSA)*. American National Standards Institute, Washington, DC, United States of America.

[ANSI, 2011] ANSI (2011). *ANSI X9.63: Public Key Cryptography for the Financial Services Industry - Key Agreement and Key Transport Using Elliptic Curve Cryptography*. American National Standards Institute, Washington, DC, United States of America.

[Antonopoulos, 2014] Antonopoulos, A. M. (2014). *Mastering Bitcoin: Unlocking Digital Crypto-Currencies*. O'Reilly Media, Inc., Sebastopol, CA, United States Of America, 1st edition.

[Ball, 2011] Ball, L. (2011). *Money, Banking and Financial Markets*. Worth Publishers, New York,NY, United States Of America, 2nd edition.

[Barber et al., 2012] Barber, S., Boyen, X., Shi, E., and Uzun, E. (2012). Bitter to Better – How to Make Bitcoin a Better Currency. In Keromytis, A., editor, *Financial Cryptography and Data Security*, volume 7397 of *Lecture Notes in Computer Science*, pages 399–414. Springer Berlin Heidelberg.

[Barlow, 1996] Barlow, J. P. (1996). A declaration of the independence of cyberspace. `https://projects.eff.org/~barlow/Declaration-Final.html`. Consultado el 7 de diciembre de 2015.

[Baumann et al., 2014] Baumann, A., Fabian, B., and Lischke, M. (2014). Exploring the Bitcoin Network. WEBIST (1), 369-374(2014).

[Beikverdi and Song, 2015] Beikverdi, A. and Song, J. (2015). Trend of centralization in Bitcoin's distributed network. In *SNPD 2015 16th IEEE/ACIS International Conference*, pages 1–6.

[Belinky et al., 2015] Belinky, M., Rennick, E., and Veitch, A. (2015). The Fintech 2.0 Paper: rebooting financial services. Report, Oliver Wyman, Anthemis Group and Santander Innoventures.

[Biryukov and Pustogarov, 2015] Biryukov, A. and Pustogarov, I. (2015). Bitcoin over Tor isn't a Good Idea. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 122–134.

[Blake et al., 2005] Blake, I., Seroussi, G., Smart, N., and Cassels, J. W. S. (2005). *Advances in Elliptic Curve Cryptography (London Mathematical Society Lecture Note Series)*. Cambridge University Press, New York, NY, United States of America.

[Blake-Wilson et al., 2006] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and Moeller, B. (2006). RFC 4492: Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS). Request for Comments 4492, The Internet Society.

[Blundell-Wignall, 2014] Blundell-Wignall (2014). The Bitcoin Question. *OECD Working Papers on Finance, Insurance and Private Pensions*, (37).

[Bonneau et al., 2015] Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J., and Felten, E. (2015). SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 104–121.

[Bos et al., 2014] Bos, J. W., Halderman, J. A., Heninger, N., Moore, J., Naehrig, M., and Wustrow, E. (2014). Elliptic Curve Cryptography in Practice. In *Financial Cryptography and Data Security*. Springer.

[Brito and Castillo, 2013] Brito, J. and Castillo, A. (2013). Bitcoin: A Primer for Policymakers. *Mercatus Center: Geroge Mason University*.

[Buchmann et al., 2013] Buchmann, J. A., Karatsiolis, E., and Wiesmaier, A. (2013). *Introduction to Public Key Infrastructures*. Springer-Verlag Berlin Heidelberg, Berlin, Germany, 1st edition.

[CAB, 2015] CAB (2015). *Baseline Requirements Certificate Policy for the Issuance and Management of Publicly-Trusted Certificates*. CA/Browser Forum. Version 1.3.1.

[Callas et al., 2007] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and Thayer, F. (2007). *RFC 4880: OpenPGP Message Format*. Internet Engineering Task Force.

[CCITT, 1991] CCITT (1991). *Recommendation X.800*. International Telegraph and Telephone Consultative Committee - International Telecommunication Union, Geneva, Switzerland.

[Chaum, 1983] Chaum, D. (1983). Blind Signatures for Untraceable Payments. In Chaum, D., Rivest, R. L., and Sherman, A. T., editors, *Advances in Cryptology — CRYPTO '82*, pages 199–203. Springer US.

[Chaves et al., 2008] Chaves, R., Kuzmanov, G., Sousa, L., and Vassiliadis, S. (2008). Cost-Efficient SHA Hardware Accelerators. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 16(8):999–1008.

[Comodo, 2011] Comodo (2011). Comodo Report of Incident - Comodo detected and thwarted an intrusion on 26-MAR-2011. Consultado el 19 de noviembre de 2015 en `https://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html`.

[Diffie and Hellman, 1976] Diffie, W. and Hellman, M. (1976). New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654.

[Dingledine et al., 2004] Dingledine, R., Mathewson, N., and Syverson, P. (2004). Tor: The Second-generation Onion Router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 21–21, Berkeley, CA, USA. USENIX Association.

[Douceur, 2002] Douceur, J. (2002). The Sybil Attack. In Druschel, P., Kaashoek, F., and Rowstron, A., editors, *Peer-to-Peer Systems*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260. Springer Berlin Heidelberg.

[ECB, 1998] ECB (1998). *Report on Electronic Money*. European Central Bank, Frankfurt, Germany.

[Eskandari et al., 2015] Eskandari, S., Barrera, D., Stobert, E., and Clark, J. (2015). A First Look at the Usability of Bitcoin Key Management. In *NDSS Workshop on Usable Security (USEC)*.

[Farrell, 2011] Farrell, S. (2011). Not Reinventing PKI Until We Have Something Better. *IEEE Internet Computing*, 15(5):95–98.

[Fox-IT, 2011] Fox-IT (2011). Black Tulip: Report of the investigation into the DigiNotar Certificate Authority breach. Consultado el 19 de noviembre de 2015 en `https://www.rijksoverheid.nl/bestanden/documenten-en-publicaties/rapporten/2012/08/13/black-tulip-update/black-tulip-update.pdf`.

[Fromknecht et al., 2014] Fromknecht, C., Velicanu, D., and Yakoubov, S. (2014). A Decentralized Public Key Infrastructure with Identity Retention. Cryptology ePrint Archive, Report 2014/803.

[Gallant et al., 2001] Gallant, R., Lambert, R., and Vanstone, S. (2001). Faster Point Multiplication on Elliptic Curves with Efficient Endomorphisms. In Kilian, J., editor, *Advances in Cryptology — CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 190–200. Springer Berlin Heidelberg.

[Garman et al., 2013] Garman, C., Green, M., and Miers, I. (2013). Decentralized Anonymous Credentials. Cryptology ePrint Archive, Report 2013/622.

[Gervais et al., 2014] Gervais, A., Karame, G., Capkun, V., and Capkun, S. (2014). Is Bitcoin a Decentralized Currency? *Security Privacy, IEEE*, 12(3):54–60.

[Gilbert and Handschuh, 2004] Gilbert, H. and Handschuh, H. (2004). Security Analysis of SHA-256 and Sisters. In Matsui, M. and Zuccherato, R., editors, *Selected Areas in Cryptography*, volume 3006 of *Lecture Notes in Computer Science*, pages 175–193. Springer Berlin Heidelberg.

[Gutman, 2002] Gutman, P. (2002). PKI: It's Not Dead, Just Resting. *Computer*, 35(8):41–49.

[Guttman and Roback, 1995] Guttman, B. and Roback, E. A. (1995). *SP 800-12. An Introduction to Computer Security: The NIST Handbook*. National Institute of Standards and Technology, Gaithersburg, MD, United States of America.

[Hankerson et al., 2003] Hankerson, D., Menezes, A. J., and Vanstone, S. (2003). *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., Secaucus, NJ, United States of America.

[Housley et al., 2002] Housley, R., Polk, W., Ford, W., and Solo, D. (2002). *RFC 3280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. Internet Engineering Task Force.

[IEEE, 2009] IEEE (2009). *IEEE Std 1363.2-2008: IEEE Standard Specifications for Password-Based Public-Key Cryptographic Techniques*. Institute of Electrical and Electronics Engineers, New York, NY, United States of America.

[ISC, 2011] ISC (2011). Can I have a TXT or SPF record longer than 255 characters? `https://kb.isc.org/article/AA-00356/0/Can-I-have-a-TXT-or-SPF-record-longer-than-255-characters.html`. Internet Systems Consortium Knowledge Base.

[ISO, 2002] ISO (2002). *ISO/IEC 15946-2:2002 Information technology – Security techniques – Cryptographic techniques based on elliptic curves – Part 2: Digital signatures*. International Organization for Standardization, Geneva, Switzerland.

[ISO, 2006] ISO (2006). *ISO/IEC 18033-2:2006 Information technology – Security techniques – Encryption algorithms – Part 2: Asymmetric ciphers*. International Organization for Standardization, Geneva, Switzerland.

[(ISRG), 2014] (ISRG), I. S. R. G. (2014). Let's Encrypt. Consultado el 19 de noviembre de 2015 en `https://www.letsencrypt.org/`.

[ITU, 1999] ITU (1999). *Recommendation F.400/X.400: Message handling system and service overview*. International Telecommunication Union.

[ITU, 2012a] ITU (2012a). *Recommendation X.500: Information technology - Open Systems Interconnection - The Directory: Overview of concepts, models and services*. International Telecommunication Union.

[ITU, 2012b] ITU (2012b). *Recommendation X.509: Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks*. International Telecommunication Union.

[ITU, 2015] ITU (2015). *Recommendation X.680: Information Technology — Abstract Syntax Notation One (ASN.1): Specification of Basic Notation*. International Telecommunication Union.

[Kaminsky, 2011] Kaminsky, D. (2011). Black Ops of TCP/IP 2011. Las Vegas, NV. BlackHat 2011.

[Karame et al., 2012] Karame, G. O., Androulaki, E., and Capkun, S. (2012). Double-spending Fast Payments in Bitcoin. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 906–917, New York, NY, USA. ACM.

[Koblitz, 1987] Koblitz, N. (1987). Elliptic Curve Cryptosystems. *Mathematics of Computation*, 48(177):203–209.

[Kohnfelder, 1978] Kohnfelder, L. M. (1978). *Towards a Practical Public-key Cryptosystem*. Bachelor of Science Thesis. Massachusetts Institute of Technology, Cambridge, MA, United States Of America.

[Kondor et al., 2014] Kondor, D., Pósfai, M., Csabai, I., and Vattay, G. (2014). Do the Rich Get Richer? An Empirical Analysis of the Bitcoin Transaction Network. *PLoS ONE*, 9(2).

[Langley, 2013] Langley, A. (2013). Further improving digital certificate security. Consultado el 19 de noviembre de 2015 en `http://googleonlinesecurity.blogspot.com/2013/12/further-improving-digital-certificate.html`.

[Langley, 2015] Langley, A. (2015). Maintaining digital certificate security. Consultado el 19 de noviembre de 2015 en `http://googleonlinesecurity.blogspot.com/2015/03/maintaining-digital-certificate-security.html`.

[Lee, 2013] Lee, T. B. (2013). An Illustrated History Of Bitcoin Crashes. *Forbes*. http://www.forbes.com/sites/timothylee/2013/04/11/an-illustrated-history-of-bitcoin-crashes/.

[Litke and Stewart, 2014] Litke, P. and Stewart, J. (2014). Cryptocurrency-Stealing Malware Landscape. Technical report, Dell SecureWorks, Inc., Atlanta, GA, USA. http://www.secureworks.com/cyber-threat-intelligence/threats/cryptocurrency-stealing-malware-landscape/.

[Martínez Silva, 2005] Martínez Silva, G. (2005). *Diseño e Implementación de una Autoridad Certificadora en Plataformas Móviles*. Tesis de maestría. Instituto Tecnológico y de Estudios Superiores de Monterrey, Distrito Federal, México.

[Martínez-Silva et al., 2007] Martínez-Silva, G., Rodríguez-Henríquez, F., Cruz-Cortés, N., and Ertaul, L. (2007). On the generation of X.509v3 certificates with biometric information. *Proceedings of The 2007 International Conference on Security and Management, SAM'07*, pages 52–57.

[Medvinsky and Neuman, 1993] Medvinsky, G. and Neuman, C. (1993). NetCash: A Design for Practical Electronic Currency on the Internet. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS '93, pages 102–106, New York, NY, USA. ACM.

[Meiklejohn et al., 2013] Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G. M., and Savage, S. (2013). A Fistful of Bitcoins: Characterizing Payments Among Men with No Names. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, pages 127–140, New York, NY, USA. ACM.

[Menezes et al., 1996] Menezes, A. J., Vanstone, S. A., and Oorschot, P. C. V. (1996). *Handbook of Applied Cryptography*. Discrete Mathematics and its Applications. CRC Press, Inc., Boca Raton, FL, United States of America.

[Merkle, 1988] Merkle, R. (1988). A digital signature based on a conventional encryption function. In Pomerance, C., editor, *Advances in Cryptology — CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 369–378. Springer Berlin Heidelberg.

[Merkle, 1978] Merkle, R. C. (1978). Secure Communications over Insecure Channels. *Communications of the ACM*, 21(4):294–299.

[Microsoft, 2015] Microsoft (2015). Microsoft Security Advisory 3046310. Consultado el 19 de noviembre de 2015 en `https://technet.microsoft.com/en-us/library/security/3046310.aspx`.

[Miller and LaViola Jr, 2014] Miller, A. and LaViola Jr, J. J. (2014). Anonymous Byzantine Consensus from Moderately-Hard Puzzles: A Model for Bitcoin. Technical report, University of Central Florida, Orlando, FL, USA.

[Miller, 1986] Miller, V. S. (1986). Use of Elliptic Curves in Cryptography. In *Lecture Notes in Computer Sciences; 218 on Advances in cryptology—CRYPTO 85*, pages 417–426, New York, NY, USA. Springer-Verlag New York, Inc.

[Mishkin, 2004] Mishkin, F. (2004). *The Economics of Money, Banking and Financial Markets*. Pearson Education, New York,NY, United States Of America, 7th edition.

[Mochón, 1995] Mochón, F. (1995). *Principios de economía*. McGraw Hill, Madrid, España.

[Mockapetris, 1987a] Mockapetris, P. (1987a). RFC 1034: DOMAIN NAMES - CONCEPTS AND FACILITIES. Request for Comments 1034, Internet Engineering Task Force.

[Mockapetris, 1987b] Mockapetris, P. (1987b). RFC 1035: DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. Request for Comments 1035, Internet Engineering Task Force.

[Moore and Christin, 2013] Moore, T. and Christin, N. (2013). Beware the Middleman: Empirical Analysis of Bitcoin-Exchange Risk. In Sadeghi, A.-R., editor, *Financial Cryptography and Data Security*, volume 7859 of *Lecture Notes in Computer Science*, pages 25–33. Springer Berlin Heidelberg.

[Nakamoto, 2008] Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. `https://bitcoin.org/bitcoin.pdf`. consultado en 24 de Noviembre de 2014.

[Neuman and Medvinsky, 1995a] Neuman, B. and Medvinsky, G. (1995a). NetCheque, NetCash, and the Characteristics of Internet Payment Services. MIT Workshop on Internet Economics.

[Neuman and Medvinsky, 1995b] Neuman, B. and Medvinsky, G. (1995b). Requirements for network payment: the NetCheque perspective. In *Compcon '95.'Technologies for the Information Superhighway', Digest of Papers.*, pages 32–36.

[NIST, 2004] NIST (2004). *FIPS 199, Standards for Security Categorization of Federal Information and Information Systems*. National Institute of Standards and Technology, Gaithersburg, MD, United States of America.

[NIST, 2012] NIST (2012). *FIPS PUB 180-4 FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION: Secure Hash Standard (SHS)*. National Institute of Standards and Technology, Gaithersburg, MD, United States of America.

[NIST, 2013] NIST (2013). *FIPS PUB 186-4 FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION: Digital Signature Standard (DSS)*. National Institute of Standards and Technology, Gaithersburg, MD, United States of America.

[Ober et al., 2013] Ober, M., Katzenbeisser, S., and Hamacher, K. (2013). Structure and Anonymity of the Bitcoin Transaction Graph. *Future Internet*, 5(2):237.

[OECD, 2002] OECD (2002). *OECD Guidelines for the Security of Information Systems and Networks: Towards a Culture of Security*. Organisation for Economic Co-operation and Development, Paris, France.

[Okamoto and Ohta, 1992] Okamoto, T. and Ohta, K. (1992). Universal Electronic Cash. In Feigenbaum, J., editor, *Advances in Cryptology — CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 324–337. Springer Berlin Heidelberg.

[O'Mahony et al., 2001] O'Mahony, D., Peirce, M., and Tewari, H. (2001). *Electronic Payment Systems for E-Commerce*. Artech House, Norwood, MA, USA, 2nd edition.

[Ondrus and Pigneur, 2009] Ondrus, J. and Pigneur, Y. (2009). Near field communication: an assessment for future payment systems. *Information Systems and e-Business Management*, 7(3):347–361.

[Paar and Pelzl, 2010] Paar, C. and Pelzl, J. (2010). *Understanding Cryptography: A Textbook for Students and Practitioners*. Springer-Verlag, Berlin, Germany.

[Rainer Böhme, 2015] Rainer Böhme, Nicolas Christin, B. E. T. M. (2015). Bitcoin: Economics, Technology, and Governance. *The Journal of Economic Perspectives*, 29(2):213–238.

[Reid and Harrigan, 2013] Reid, F. and Harrigan, M. (2013). An Analysis of Anonymity in the Bitcoin System. In Altshuler, Y., Elovici, Y., Cremers, A. B., Aharony, N., and Pentland, A., editors, *Security and Privacy in Social Networks*, pages 197–223. Springer New York.

[Ron and Shamir, 2013] Ron, D. and Shamir, A. (2013). Quantitative Analysis of the Full Bitcoin Transaction Graph. In Sadeghi, A.-R., editor, *Financial Cryptography and Data Security*, volume 7859 of *Lecture Notes in Computer Science*, pages 6–24. Springer Berlin Heidelberg.

[Ron and Shamir, 2014] Ron, D. and Shamir, A. (2014). How Did Dread Pirate Roberts Acquire and Protect his Bitcoin Wealth? In Böhme, R., Brenner, M., Moore, T., and Smith, M., editors, *Financial Cryptography and Data Security*, volume 8438 of *Lecture Notes in Computer Science*, pages 3–15. Springer Berlin Heidelberg.

[Salmon, 2013] Salmon, F. (2013). The Bitcoin Bubble and the Future of Currency. *Medium*. https://medium.com/@felixsalmon/the-bitcoin-bubble-and-the-future-of-currency-2b5ef79482cb.

[Schneier, 1995] Schneier, B. (1995). *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., New York, NY, United States of America, 2nd edition.

[SECG, 2009] SECG (2009). *SEC 1: Elliptic Curve Cryptography, Version 2.0*. Certicom Research, Mississauga, ON, Canada.

[Shirey, 2007] Shirey, R. (2007). RFC 4949 (Informational): Internet Security Glossary, Version 2. Request for Comments 4949, Internet Engineering Task Force.

[Sleevi, 2015] Sleevi, R. (2015). Sustaining Digital Certificate Security. Consultado el 19 de noviembre de 2015 en `https://googleonlinesecurity.blogspot.mx/2015/10/sustaining-digital-certificate-security.html`.

[Somogyi and Eijdenberg, 2015] Somogyi, S. and Eijdenberg, A. (2015). Improved Digital Certificate Security. Consultado el 19 de noviembre de 2015 en `https://googleonlinesecurity.blogspot.mx/2015/09/improved-digital-certificate-security.html`.

[Stallings, 2013] Stallings, W. (2013). *Cryptography and Network Security: Principles and Practice*. Pearson Education, New York, NY, United States of America, 6th edition.

[Stavroulakis and Stamp, 2010] Stavroulakis, P. and Stamp, M. (2010). *Handbook of Information and Communication Security*. Springer-Verlag, Berlin, Germany.

[Steiner et al., 1988] Steiner, J. G., Neuman, C., and Schiller, J. I. (1988). Kerberos: An Authentication Service for Open Network Systems. In *USENIX Conference Proceedings*, pages 191–202.

[Swan, 2015] Swan, M. (2015). *Blockchain*. O'Reilly Media, Inc., Sebastopol, CA, United States Of America, 1st edition.

[Symantec, 2015a] Symantec (2015a). Incident Report 1. Consultado el 19 de noviembre de 2015 en `https://www-secure.symantec.com/connect/sites/default/files/TestCertificateIncidentReportOwnedDomains.pdf`.

[Symantec, 2015b] Symantec (2015b). Incident Report 2. Consultado el 19 de noviembre de 2015 en `https://www-secure.symantec.com/connect/sites/default/files/TestCertificateIncidentReportUnregisteredv2.pdf`.

[Symantec, 2015c] Symantec (2015c). Test Certificates Incident Final Report. Consultado el 19 de noviembre de 2015 en `https://www-secure.symantec.com/connect/sites/default/files/Test_Certificates_Incident_Final_Report_10_13_2015v3b.pdf`.

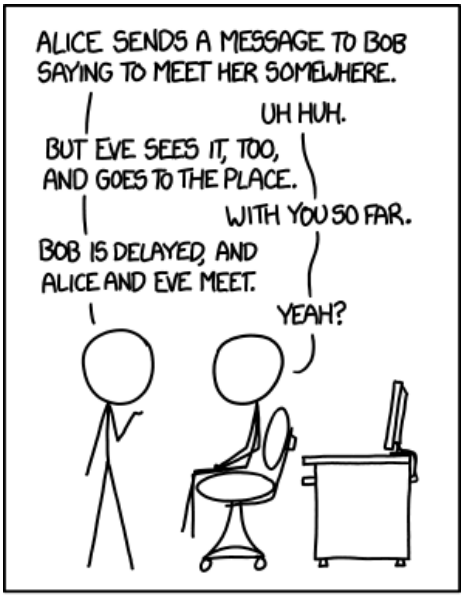
[Tschorsch and Scheuermann, 2015] Tschorsch, F. and Scheuermann, B. (2015). Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies. Cryptology ePrint Archive, Report 2015/464. `http://eprint.iacr.org/`.

[Vacca, 2013] Vacca, J. R. (2013). *Computer and Information Security Handbook*. Morgan Kaufmann Publishers Inc., San Francisco, CA, United States of America, 2nd edition.

[Van Alstyne, 2014] Van Alstyne, M. (2014). Why Bitcoin Has Value. *Commun. ACM*, 57(5):30–32.

[Vasek and Moore, 2015] Vasek, M. and Moore, T. (2015). There's No Free Lunch, Even Using Bitcoin: Tracking the Popularity and Profits of Virtual Currency Scams. In Böhme, R. and Okamoto, T., editors, *Financial Cryptography and Data Security*, volume 8975 of *Lecture Notes in Computer Science*, pages 44–61. Springer Berlin Heidelberg.

[Zimmermann, 1995] Zimmermann, P. (1995). *PGP Source Code and Internals*. MIT Press, Cambridge, MA, United States of America, 1st edition.



Fuente: `http://xkcd.com/1323/`