UNIVERSIDAD DE GRANADA

TRABAJO FIN DE MASTER

INGENIERÍA DE TELECOMUNICACIÓN

Titulo del Proyecto

Subtitulo del Proyecto

Autor

Nombre Apellido1 Apellido2 (alumno)

Directores

Nombre Apellido1 Apellido2 (tutor1)

Nombre Apellido1 Apellido2 (tutor2)

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, mes de 201

Título del proyecto

Subtítulo del proyecto.

Autor
Nombre Apellido1 Apellido2 (alumno)

Directores
Nombre Apellido1 Apellido2 (tutor1)
Nombre Apellido1 Apellido2 (tutor2)

Título del Proyecto: Subtítulo del proyecto

Nombre Apellido1 Apellido2 (alumno)

Palabras clave: palabra_clave1, palabra_clave2, palabra_clave3, ...

Resumen

Poner aquí el resumen.

Project Title: Project Subtitle

First name, Family name (student)

Keywords: Keyword1, Keyword2, Keyword3, ...

Abstract

Write here the abstract in English.

Yo, **Nombre Apellido1 Apellido2**, alumno de la titulación TITULACIÓN de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI XXXXXXXXX, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Master en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Nombre Apellido1 Apellido2

Granada a X de mes de 201 .

D. **Nombre Apellido1 Apellido2 (tutor1)**, Profesor del Área de XXXX del Departamento YYYY de la Universidad de Granada.

D. **Nombre Apellido1 Apellido2 (tutor2)**, Profesor del Área de XXXX del Departamento YYYY de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Título del proyecto, Subtítulo del proyecto***, ha sido realizado bajo su supervisión por **Nombre Apellido1 Apellido2 (alumno)**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a X de mes de 201 .

Los directores:

Nombre Apellido1 Apellido2 (tutor1) **Nombre Apellido1 Apellido2 (tutor2)**

Agradecimientos

Poner aquí agradecimientos...

Capítulo 1

Introducción

1.1. Motivación del proyecto

En el año 1994, tras una reunión en Sillicon Valley, Timothy C. May publica el manifiesto cripto-anarquista [31] que da origen al movimiento Cyberpunk. Este movimiento social es el origen de una revolución de carácter libertaria cuyo principal objetivo es reducir la intervención del Estado a su mínima expresión. Se consideraba que la protección brindada por el Estado en la relación y acuerdos entre individuos, podía ser brindada de una manera más libertaria y eficiente por la tecnología. Dicho movimiento siembra las premisas de un proyecto que surge años después, el proyecto Bitcoin.

El 1 de noviembre de 2008 se publica un *whitepaper* titulado "Bitcoin: A Peer-to-Peer Electronic Cash System" [33]. El autor (o autores) del *whitepaper*, que se esconde bajo el alias Satoshi Nakamoto, decide publicarlo en una lista de correo sobre criptografía. Este es uno de los primeros documentos en los que se menciona el término Blockchain. Se describe como una tecnología creada con el propósito de llevar a cabo el proyecto de Bitcoin. Blockchain surge de la ingeniosa combinación de distintas técnicas creadas años atrás en el campo de la informática y la criptografía. Apoyándose en los ideales del movimiento Cyberpunk, se describe al Bitcoin como una moneda digital gobernada por unas reglas criptográficas que controlan su gestión y emisión. La tecnología blockchain garantiza el correcto funcionamiento de los aspectos clave de las criptomonedas: la puesta en circulación de la moneda y el mecanismo de gasto que garantiza que una moneda no pueda ser gastada varias veces. Anteriores monedas digitales habían intentado resolver estos dos problemas sin éxito. El proyecto Bitcoin, cuya base tecnológica es blockchain, es la primera criptomoneda capaz de resolverlos de manera totalmente descentralizada.

Como cualquier otra nueva tecnología, Blockchain debe pasar por todas

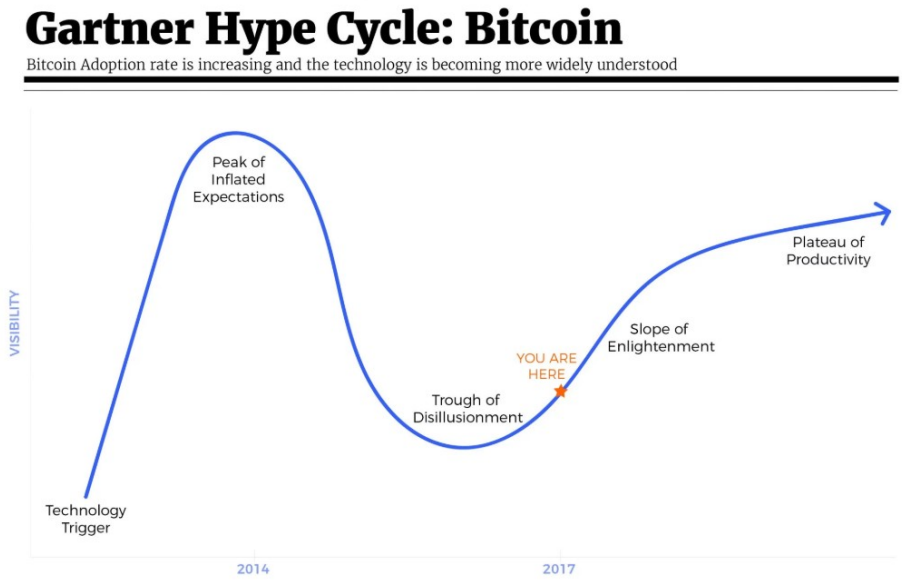

Figura 1.1: Curva Gartner de Bitcoin [10].

las fases de adopción que se observan en la figura XXXXXXXXXXX. Durante los primero años, el término blockchain está ligado directamente con el proyecto Bitcoin, ambos términos se conciben como un todo.

El gran abanico de posibilidades que parece ofrecer blockchain a parte de Bitcoin hace que se llegue al pico de expectación de la tecnología. Aunque se empieza a entender que blockchain y Bitcoin no son la misma cosa, el interés por blockchain se sigue manifestando de la manera más accesible por cualquier persona, comprando bitcoin. No se debe confundir la curva de la figura XXXXXX con la del precio de Bitcoin. La fase de desilusión de la tecnología no se corresponde con la bajada de precio de Bitcoin. A medida que se estudia a fondo blockchain y todos sus posibles casos de uso afloran todos los inconvenientes y problemas de la tecnología. El principal problema es la escalabilidad. Este hace que se empiece a pensar que la tecnología que parecía poderse aplicar a un sinfín de usos ahora no se pueda aplicar a nada. En la fase "pendiente de la iluminación.[en] la que se encuentra blockchain actualmente se trata de solucionar todos los problemas que se han descubierto en la anterior fase para poder aplicar la tecnología a los sectores en los que de verdad aporta valor. Es así como se consigue llegar a la "meseta de productividad".

1.2. Objetivos

En el apartado ANTERIORXXXXXXXXXXXXXXXXXXX se describen algunos de los hitos más relevantes del origen de las criptomonedas y blockchain. La actual fase en la que se encuentra la tecnología motiva la realización del presente trabajo. En este apartado se enumerarán los objetivos concretos del trabajo con el que se pretende contribuir a la mejora de la escalabilidad de blockchain.

A continuación se enumeran los objetivos del presente trabajo, partiendo desde los más simples y concretos hasta llegar a los objetivos que constituyan el sistema que se quiere desarrollar:

1. Mecanismo con el cuál comparar dos relojes. Para ello se necesita comparar cada reloj por separado con una misma referencia temporal mutua para ambos.
2. Generación de ficheros RINEX mediante un dispositivo de posicionamiento GPS.
3. A partir de los ficheros RINEX y haciendo uso del software R2CGGTTS generar el fichero con el que obtener la trazabilidad temporal. Al fichero se le denomina CGGTTS.
4. Estudiar los distintos fabricantes de procesadores que ofrecen Trusted Execution Environment (TEE). A partir del estudio elegir el sistema empotrado que se usará.
5. Desarrollar una aplicación de ejemplo que se ejecute en uno de estos enclaves seguros de ejecución.
6. Desarrollar un sistema operativo seguro y confiable que constituya el ambiente de ejecución seguro en el que se ejecutará la aplicación desarrollada.
7. Ejecución de un protocolo de sincronización de relojes que proporcione precisión al sistema. En concreto PTPd (Precision Time Protocol daemon).
8. La aplicación será la encargada de realizar el sellado temporal de información. En concreto, realizará el sellado temporal de los *hashes* los bloques de blockchain y del fichero CGGTTS.
9. Desarrollar un protocolo blockchain alternativo que haga uso del sistema de sellado temporal confiable.
10. Evaluación de la mejora de escalabilidad conseguida con el protocolo desarrollado.

Los objetivos que se acaban de enumerar buscan combinar el potencial de dos tecnologías como son la Trazabilidad temporal y los Trusted Execution Environment. Dicha combinación permite sacar más partido de la tecnología blockchain haciéndola más eficiente y escalable.

1.3. Materiales y métodos

Para la realización del presente trabajo se necesitan una serie de materiales (hardware y software) y seguir una serie de metodologías con las que conseguir los objetivos marcados. Se distinguen 3 tipos de objetivos, por lo que cada uno necesitará de sus materiales y métodos

1.3.1. Trazabilidad de tiempo

Para conseguir los objetivos relacionados con la trazabilidad de tiempo se hace uso del siguiente dispositivo y librerías de código.

1.3.1.1. Dispositivo U-blox

La empresa U-blox fabrica una serie de dispositivos de posicionamiento GPS (entre otros propósitos). Haciendo uso de uno de sus dispositivos y del programa u-center se consiguen generar ficheros en formato ublox que contienen los mensajes enviados por los satélites GPS y recibidos por el dispositivo. A partir de dichos ficheros se generan los ficheros RINEX. Para este trabajo se ha hecho uso del dispositivo U-blox EVK-M8F [23].

1.3.1.2. RTKLIB

Se trata de un conjunto de librerías y aplicaciones de código libre destinadas al posicionamiento GNSS. Soporta multitud de dispositivos receptores, formatos, versiones, etc. Las aplicaciones que hacen uso de las librerías permiten entre otras cosas generar ficheros RINEX (GPS, GLONASS, Galileo, ...) a partir de ficheros ublox. Se hace uso de la aplicación CONVBIN que usa la librería RTKCONV [21].

1.3.1.3. R2CGGTTS

RINEX to CGGTTS es una herramienta software [8] creada por el Royal Observatory of Belgium capaz de crear ficheros CGGTTS a partir de ficheros RINEX.

1.3.2. Trusted Execution Environment (TEE)

Para la seguridad y confiabilidad del sistema se necesita un TEE. Varios fabricantes de procesadores ofrecen esta tecnología, cada uno con sus librerías y para sus dispositivos.

1.3.2.1. ARM TrustZone

El fabricante ARM es uno de los que ofrece esta tecnología. ARM la denomina TrustZone [28]. La mayoría de sus familias de procesadores actuales soportan el uso de TrustZone. En función de la familia del procesador se tendrán unas opciones u otras.

1.3.2.2. Zedboard

Zedboard es un kit de desarrollo para desarrolladores interesados en usar Xilinx Zynq-7000 All Programmable SoC [27]. La tarjeta Zedboard contiene un procesador Dual ARM Cortex-A9 MPCore, uno de los que soportan TrustZone. Para el desarrollo en Zedboard se hace uso del software de Xilinx Vivado y de la documentación de Xilinx relacionada con la seguridad de sus tarjetas.

1.3.2.3. Raspberry Pi

Para el presente trabajo se hace uso de Raspberry Pi 3 Modelo B [18]. El principal requisito para el trabajo es su procesador. Este modelo de Raspberry monta un Broadcom BCM2837 Cortex-A53 (ARMv8). Dicha familia de ARM soporta TrustZone.

1.3.2.4. OP-TEE

Open Portable Trusted Execution Environment [14] es un proyecto de código libre que trata de llevar TEE a multitud de dispositivos. Entre los dispositivos a los que lleva TEE está la Raspberry Pi 3.

1.3.2.5. Buildroot

Buildroot es un proyecto de código libre cuyo propósito es generar sistemas Linux para sistemas empotrados de manera simple y customizable. El proyecto OP-TEE hace uso de Buildroot para generar el sistema operativo, por tanto, si se quieren añadir nuevos paquetes que Buildroot no traiga por defecto hay que hacer uso de sus herramientas [5].

1.3.2.6. Precision Time Protocol (PTP)

Uno de los paquetes que se le puede añadir al sistema operativo gracias a Buildroot es PTPd [15]. PTP daemon es una implementación del protocolo Precision Time Protocol que permite la sincronización precisa de relojes de equipos conectados mediante Ethernet.

1.3.3. Protocolo Blockchain

El objetivo del presente trabajo es aplicar el sistema desarrollado al protocolo de una red blockchain de manera que esto suponga una mejora en escalabilidad o eficiencia.

1.3.3.1. Repositorio Bitcoin

Para este objetivo no se parte de cero, se parte de un código existente al que se le hacen los correspondientes cambios. El código a utilizar es el de Bitcoin [2], sin embargo, se podrían aplicar los mismos cambios al código de otra blockchain basada en prueba de trabajo.

Tras enumerar los materiales y métodos que se utilizan en el presente trabajo, se pretende hacer ver mediante la figura XXXXXXXXXX la relación entre ellas. A priori pueden parecer tecnologías muy dispares, sin embargo, el sistema desarrollado las combina haciendo que tenga sentido su uso conjunto.

1.4. Organización de la memoria

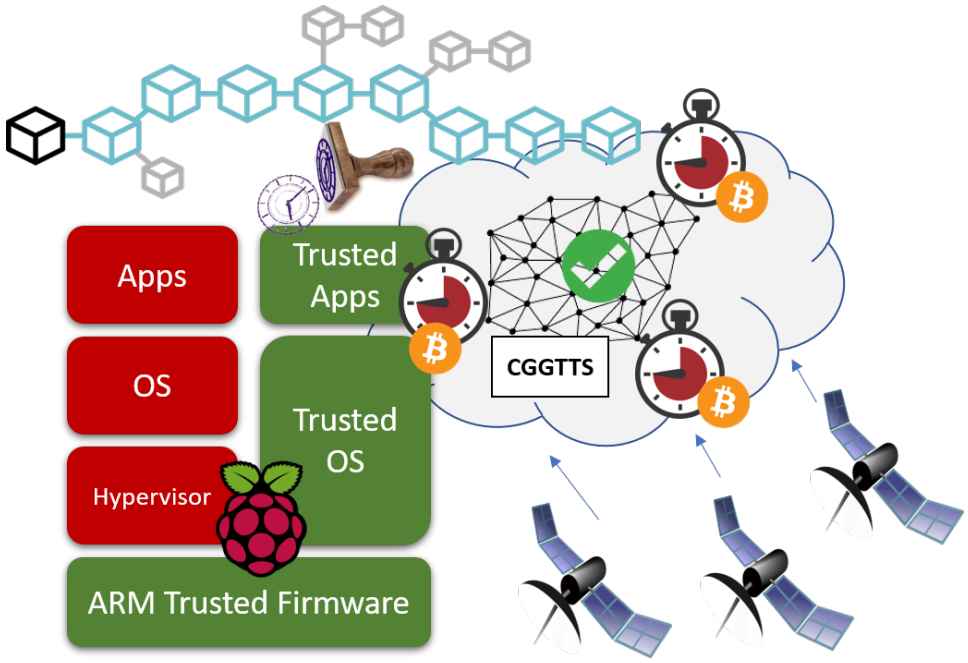

Figura 1.2: Combinación de materiales y métodos del trabajo.

Capítulo 2

Estado de la técnica

2.1. Trazabilidad de tiempo

Una de las bases sobre las que se asienta el trabajo es la trazabilidad de tiempo. Conocer la trazabilidad de un reloj a una referencia temporal universal es esencial para tener un mecanismo objetivo con el que medir la precisión de un reloj.

2.1.1. Global Navigation Satellite System (GNSS)

La GNSS se trata de un conjunto de satélites que permiten a un usuario situado en La Tierra calcular su posición exacta de manera bastante precisas (desde algunos metros hasta centímetros de precisión). Existen varios de estos sistemas de posicionamiento, entre ellos están GPS, GLONASS, Galileo, etc. Este tipo de sistemas consisten en una constelación de satélites encargados de transmitir unas determinadas señales sobre La Tierra. Los relojes que portan dichos satélites están sincronizados en base a una escala temporal definida por el sistema propio de cada constelación. Con esto se consigue que las señales emitidas por cada satélite compartan una misma referencia temporal. Esto es vital para el buen funcionamiento del sistema.

El equipo receptor de un usuario que pretende usar el sistema GNSS para determinar su posición consiste de una antena que recibe las señales emitidas por estos satélites, y un dispositivo que sea capaz de traducir estas señales recibidas en una posición. Los cálculos realizados por este dispositivo se basan en la diferencia temporal entre la emisión de la señal por parte de los satélites y la recepción por parte de la antena del usuario. Este retardo de propagación se traduce fácilmente en una distancia que separa a cada satélite de la antena, ya que la señal viaja a la velocidad de la luz.

La distancia entre un satélite y la antena no da la posición exacta de la antena, sólo informa de una esfera alrededor del satélite en la que debe

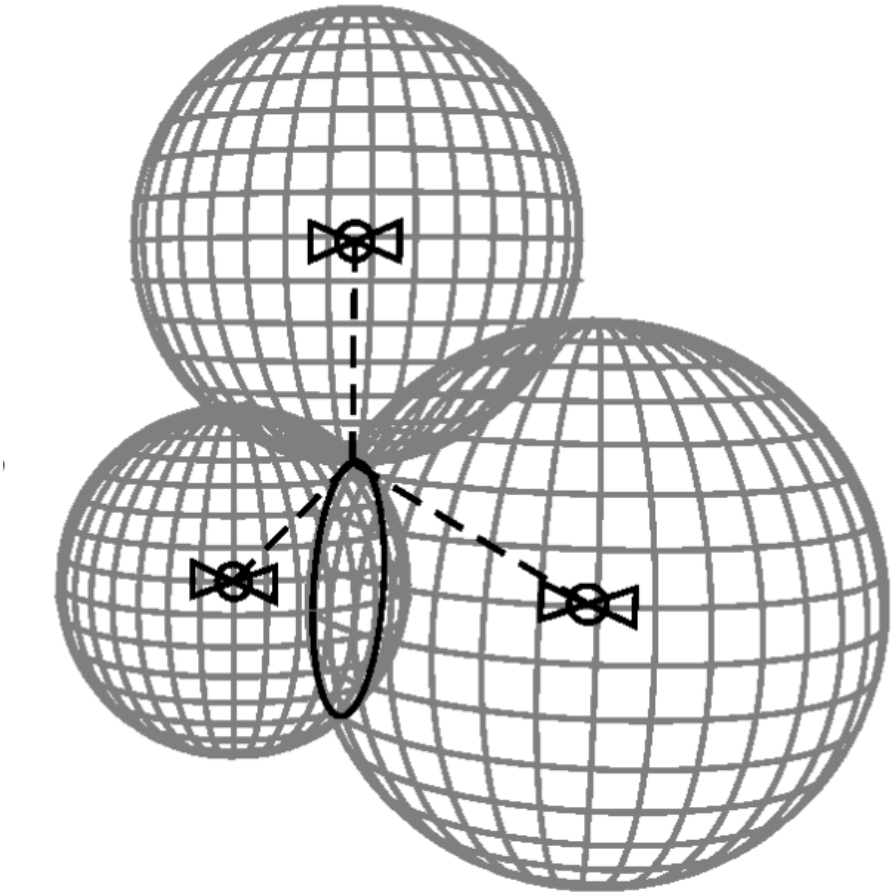

Figura 2.1: Intersección de las 2 esferas creadas por satélites [20].

encontrarse la antena. Para calcular la posición exacta se necesitan varias de estas medidas de diferentes satélites. Conociendo la distancia a la que se encuentra la antena respecto a dos satélites distintos se traduce en dos esferas distintas sobre las cuales debe localizarse la antena. Es evidente que la antena se encuentra en la intersección de estas dos esferas, que como se observa en la figura XXXXXXXx es un círculo.

La intersección de este círculo con una tercera esfera (relacionada con la distancia respecto de un satélite a la que se encuentra la antena) reducirá a 2 los posibles puntos en los que se encuentra la antena. Una cuarta esfera será necesaria para decidir cuál de esos dos puntos se corresponde con la correcta localización de la antena.

La situación descrita en la figura XXXXXXXXXXXX en la que los satélites se distribuyen espacialmente alrededor del receptor es demasia-

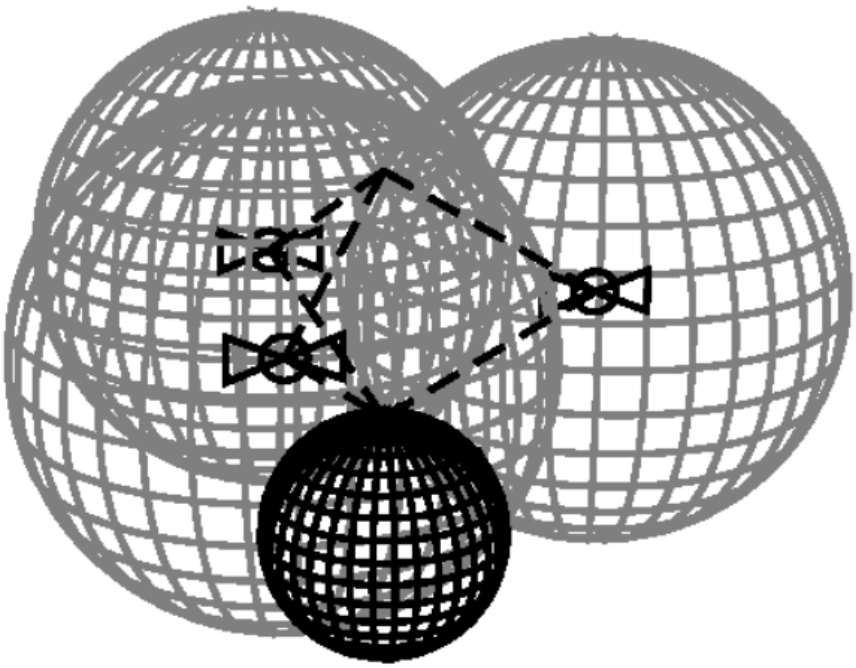

Figura 2.2: La Tierra hace de cuarta esfera [20].

do optimista. El receptor presumiblemente se situará en la superficie terrestre, por lo que solamente tendrá visibles los satélites que se encuentren sobre él. Es necesario por tanto usar la técnica observada en la figura XXXXXXXXXXXXX.

En esta situación, se tiene la intersección formada por las esferas de 3 satélites, y será la superficie de La Tierra la encargada de decidir cuál de los dos puntos de esa intersección es el correcto. La metodología es similar a la anterior, con la única diferencia de la cuarta esfera.

La descripción del funcionamiento dada del sistema GNSS se corresponde con una situación idílica, que en realidad no se corresponde con lo que ocurre en el mundo real. Existen varios problemas para los cuales se ha tenido que adaptar la metodología seguida para calcular la posición de la antena receptora.

Para poder calcular el retardo de propagación de la señal enviada por el satélite, esta va sellada con el tiempo en el que se emitió. Esto implica que el receptor debe tener un reloj preciso y sincronizado con la referencia temporal de los satélites. Esta implicación no es realista, por lo que se producen pequeños errores temporales, que a la velocidad de la luz se traducen

en grandes errores en la localización. Estos errores pueden llegar a hacer que las 4 esferas no se corten en un único punto común, por lo que son insostenibles. La solución consiste en una constante corrección del reloj del receptor que haga a las 4 esferas coincidir en un único punto.

Otro de los inconvenientes deriva del cálculo de los centros de estas esferas utilizadas para el cálculo de la localización de la antena. Esto implica que cada satélite debe ser capaz de calcular su posición exacta en el espacio, cosa que no sucede. Los satélites no son capaces de calcular su posición exactamente, lo que de nuevo se traducirá en que las 4 esferas no se corten en un único punto. Para solventar este inconveniente se sigue una estrategia distinta a la anterior. Se usarán los cálculos de tantos satélites como sea posible medir para calcular la posición de la antena de la manera más exacta posible.

Suponiendo que los anteriores problemas quedan solucionados, para que la medida de la posición sea precisa se necesitará que la señal viaje en el vacío. El retardo de propagación se ha supuesto para que el medio de transmisión sea el vacío, cosa que en el mundo real no será así. La señal atravesará algunas capas como la ionosfera y la troposfera. En estas capas de la atmósfera la velocidad será inferior a la velocidad de la luz. Se estudia como afectan las velocidades de propagación en estos medios y se realizan las correcciones pertinentes para solventar este problema.

2.1.2. Transferencia de tiempo

Cada sistema GNSS tiene su propia referencia temporal necesaria para calcular el comportamiento de los relojes contenidos en los satélites de la constelación. Esta referencia temporal es transmitida por dichos satélites en sus datos de navegación. Esta información se corresponde con la diferencia de la hora del reloj del satélite y la hora UTC. Surge entonces el concepto de transferencia o trazabilidad de tiempo consistente en la comparación de dos relojes remotos que han sido conectados a receptores GNSS. Si se conoce la localización del satélite y de la estación receptora, se puede deducir el error de sincronización con respecto a la referencia temporal del sistema GNSS. A su vez se pueden deducir diferencias temporales del reloj de diferentes laboratorios o estaciones. El funcionamiento básico del sistema usado para comparar dos relojes es el observado en la figura XXXXXXXXXXxx.

La figura XXXXXX representa una situación simplificada en la que se hace uso de un único satélite. Sin embargo, existen dos técnicas distintas en las que se hace uso de más de un satélite, que son las que realmente se llevan a cabo.

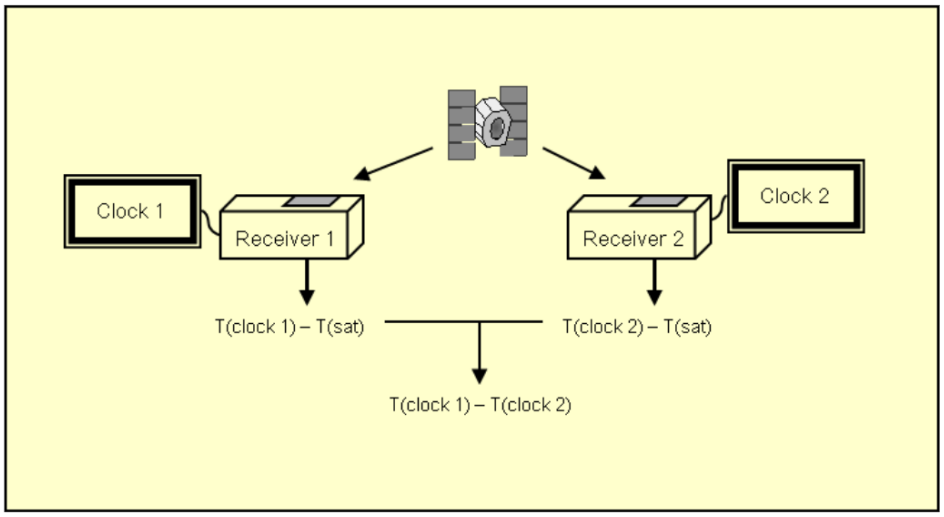

Figura 2.3: Escenario básico para la comparación de 2 relojes [20].

2.1.2.1. Satélites en Common View (CV)

Esta técnica hace uso de los satélites que son comunes durante un determinado tiempo para los dos relojes que se quieren comparar. Al ser satélites comunes se puede obtener la diferencia entre el reloj 1 y el reloj 2 en base a las medidas tomadas de cada satélite. El siguiente paso será combinar las comparaciones obtenidas con cada satélite para obtener la media del error de sincronización entre relojes. La figura XXXXXXXXXx representa la situación que se tendría al hacer uso de esta técnica.

2.1.2.2. Satélites All in view (AV)

Otra de las técnicas usada para la trazabilidad del tiempo consiste en obtener el error de sincronización respecto a la referencia temporal usada por el sistema. En este caso la referencia temporal será la hora UTC. Se hará uso de las medidas tomadas de cada satélite visto por el receptor. En los datos de navegación cada satélite transmitirá la información relacionada con el error de sincronización entre el reloj del satélite respecto a la referencia temporal del sistema. Por lo tanto, del error de sincronización entre el reloj del receptor y el del satélite se podrá deducir el error de sincronización existente entre el reloj del receptor y la referencia temporal del sistema. Cada reloj tomará estas medidas de todos los satélites que tenga a la vista como se observa en la figura XXXXXXXXXXX. Se obtiene así la media del error de sincronización de cada reloj con la referencia temporal del sistema. Como esta referencia temporal es común para todo el sistema, los dos relojes

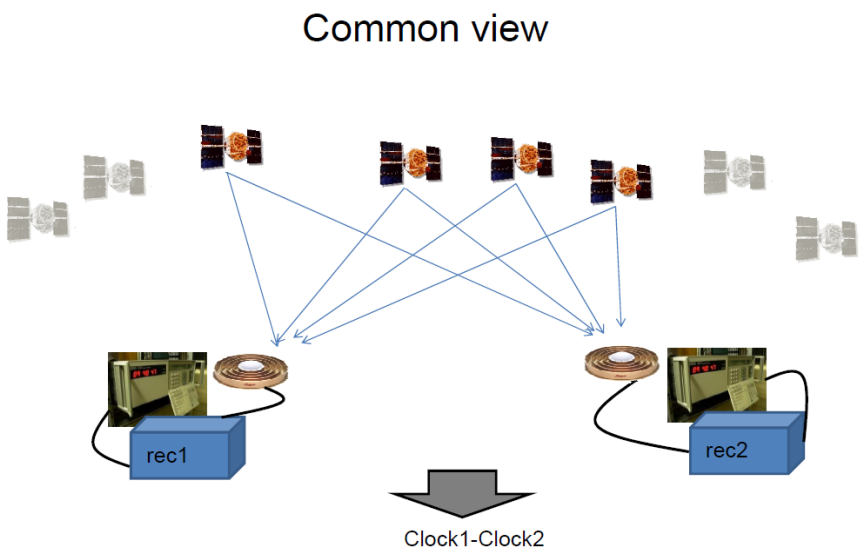

Figura 2.4: Escenario con satélites en Common View [30].

se estarán comparando con la misma referencia. De esto se podrá deducir fácilmente la comparación entre el reloj 1 y el reloj 2.

El uso de una u otra técnica dependerá principalmente de los satélites de los que se disponga. El resultado obtenido será en ambos casos el mismo, con la diferencia de la precisión. El uso de métodos como Precise Point Positioning (PPP) será uno de los principales factores determinantes para la precisión de los resultados obtenidos.

2.1.3. R2CGGTTS

El uso de la transferencia de tiempo ha sido muy extendido desde los años 80. El Bureau International des Poids et Mesures (BIPM) lo lleva utilizando desde entonces para comparar las realizaciones del tiempo UTC por parte de los laboratorios de tiempo, con el objetivo de generar el International Atomic Time (TAI). Otro de los usos más extendido consiste en el uso de la transferencia de tiempo para sincronizar relojes locales con las referencias legales de tiempo situadas en los Institutos Nacionales de Meteorología. Se hizo necesario crear un estándar con el que formatear los resultados obtenidos de la transferencia de tiempo para facilitar su uso. El estándar fue conocido como Common GPS GLONASS Time Transfer Standard (CGGTTS).

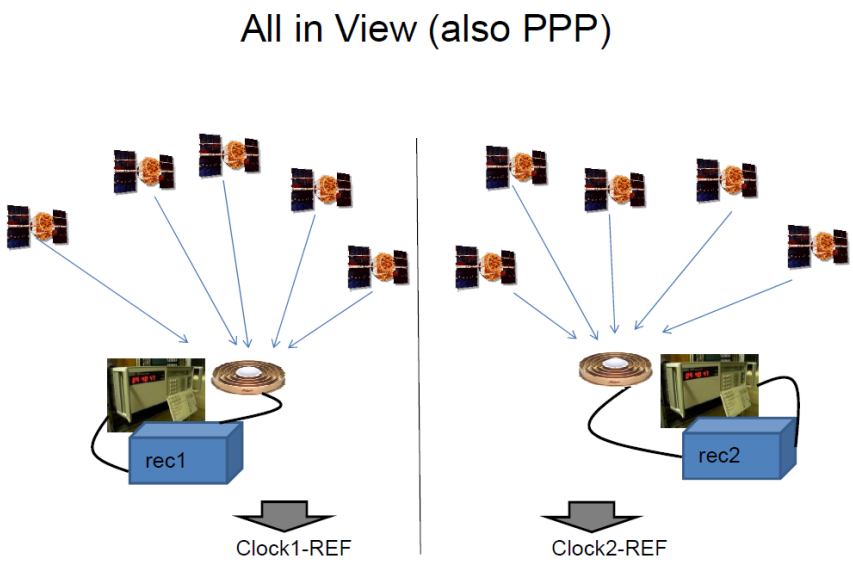

Figura 2.5: Escenario con satélites All in View [30].

Los ficheros CGGTTS pueden ser directamente producidos por el receptor GNSS. También podrán ser reconstruidos a partir de las medidas y los datos de navegación proporcionadas por el receptor en forma de fichero RINEX (Receiver INdependent EXchange). Para realizar esta tarea el Royal Observatory of Belgium (ROB) ha creado una herramienta capaz de crear ficheros CGGTTS a partir de los ficheros RINEX. Este software, denominado R2CGGTTS, es distribuido públicamente a través del servidor FTP del BIPM.

Los datos relacionados con los errores de sincronización aparecen en las dos columnas indicadas en la figura XXXXXXXXXXXXx.

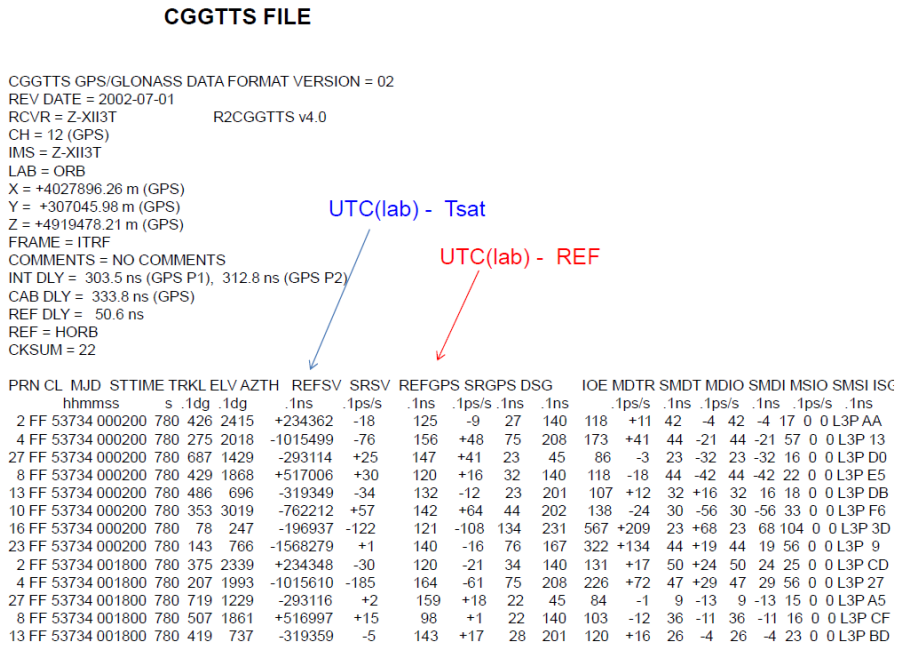

Figura 2.6: Ejemplo de fichero CGGTTS [30].

2.2. Trusted Execution Environments (TEE)

Los dispositivos embebidos cada vez manejan datos más sensibles o de más valor. Un claro ejemplo de esto son los dispositivos que manejan datos de cuentas bancarias como contraseñas. A menudo estos dispositivos cumplen otros propósitos, por lo que tienen la capacidad de descargar otro software de terceros. Esto hace que estos dispositivos se encuentren en una posición muy vulnerable. Cuanto más valioso sea el resultado de un ataque realizado con éxito a estos dispositivos, más vulnerables serán, ya que los atacantes estarán dispuestos a emplear mayor tiempo y dinero en realizar un ataque.

2.2.1. Intel Software Guard Extensions (Intel SGX)

El fabricante Intel es uno de las opciones en cuanto a TEEs. Intel denomina SGX [13] a esta tecnología que permite a los desarrolladores utilizar un entorno de ejecucion confiable a nivel de aplicación. Estos entornos se llaman enclaves y proporcionan confidencialidad e integridad, autenticación y provisión remota y una superficie de ataque pequeña.

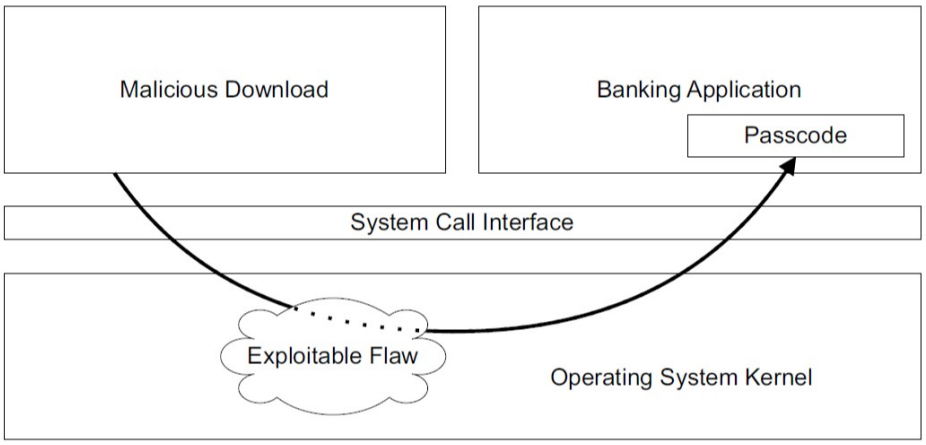

Figura 2.7: Escenario de ataque típico en hardware embebido [28].

2.2.2. AMD GuardMI Technology

La estrategia de seguridad en la que se basa AMD GuardMI Technology [1] proporciona (1) cifrado seguro y transparente de memoria, (2) arrancado seguro, (3) habilitar aplicaciones confiables y (4) un entorno seguro de producción.

2.2.3. ARM TrustZone

Entre todos los fabricantes posibles que soportan TEE, para este trabajo se utiliza ARM debido a los dispositivos elegidos. El objetivo de la tecnología ARM Trustzone consiste en conseguir que un dispositivo con sistema operativo completo sea a su vez robusto y seguro. Un sistema cuyo diseño de arquitectura hardware y software seguro sea el apropiado, podrá garantizar el correcto uso de datos sensibles. Para conseguir esto hardware y software deben diseñarse para trabajar en sintonía. Los dispositivos identificados como más seguros son aquellos que contienen medidas de seguridad desde su diseño y fabricación. Varias consideraciones de seguridad hardware solamente se podrán incluir en estas fases, añadirlas después será imposible.

La tecnología TrustZone de ARM integra varias medidas de seguridad en procesadores, buses y periféricos. Esto permitirá una gran variedad de posibilidades de arquitecturas seguras en sus dispositivos.

2.2.3.1. Vectores de ataque

Para poder estructurar una buena defensa es necesario conocer los distintos tipos de ataque que se pueden llevar a cabo.

Ataques Hack En este tipo de ataque, el atacante solamente es capaz de ejecutar un ataque software. Engloba todo tipo de virus y malware que es descargado e instalado en el dispositivo.

Ataques Shack En este tipo de ataque, el atacante suele tener acceso físico al dispositivo y dispone de un equipamiento muy básico para llevar a cabo el ataque. Con este equipamiento podrá conectarse al dispositivo vía JTAG, escanear los periféricos de entrada y salida, y mediante analizadores de red inspeccionar buses, pines y señales del sistema. A su vez podrán reprogramar espacios de memoria, reemplazar componentes del hardware, forzar buses o pines a estar en alta o baja tensión, etc.

Ataques de laboratorio Es el tipo de ataque más invasivo, ya que el atacante dispone de equipamiento de laboratorio con el que puede llevar a cabo ingeniería inversa sobre el dispositivo. Defenderse de este tipo de ataques no suele ser la estrategia más óptima. Será mucho más óptimo limitar el daño que se pueda ocasionar una vez que el dispositivo a sufrido con éxito uno de estos ataques. El objetivo consiste en hacer el ataque de laboratorio lo menos rentable posible.

2.2.3.2. Seguridad en sistemas embebidos

Los sistemas embebidos presentan diversas dificultades a la hora de hablar de seguridad. Numerosos procesadores, buses, periféricos, espacios de memoria, etc que complican la seguridad del dispositivo. La solución que haga seguro el dispositivo debe integrar todos estos componentes, ya que no será óptimo hacer seguro cada componente por separado.

Existen diversas soluciones en el mercado encargadas de la seguridad de estos dispositivos.

Módulos hardware externos de seguridad Una de las soluciones de seguridad más extendida para este tipo de dispositivos consiste en integrar un módulo externo al dispositivo en el que se confíe. Un claro ejemplo de estos son las tarjetas SIM.

Módulos hardware internos de seguridad Se trata de módulos dedicados a la seguridad del dispositivo que están integrados en el mismo. En muchos escenarios este tipo de módulos resultan más convenientes. Los ejemplos más representativos de este tipo de módulos son (1) módulos que manejan las operaciones y el almacenamiento de las tareas criptográficas y (2) módulos de procesamiento de tareas sensibles.

Virtualización software La virtualización software es un mecanismo de seguridad en el que una capa de gestión, conocida como hypervisor, separa varias plataformas software independientes. Esta separación se consigue haciendo uso del Memory Management Unit (MMU). Permite que cada uno de los softwares, que corren independientes a los demás, tengan la seguridad de no poder ser atacados por software que esté separado de este.

2.2.3.3. Seguridad TrustZone

Los principales problemas que surgen con las soluciones que existen actualmente en el mercado se deben principalmente a que estos protegen un determinado componente, dato sensible o activo, pero ignorando un gran abanico de posibilidades de ataque. Esto provoca que en ocasiones se proteja lo que no se debe, o que la protección carezca de sentido. Por ejemplo, no tiene sentido proteger el módulo que gestiona y almacena las claves de criptografía si posteriormente los datos descifrados pueden ser robados fácilmente por un atacante. Esta es la razón por la que se dice que el diseño de la seguridad debe integrar el dispositivo al completo y no solamente sus componentes por separado.

La seguridad de TrustZone se basa en el concepto de una plataforma segura y confiable, una arquitectura hardware que extiende las funcionalidades de seguridad partiendo desde su diseño y fabricación. Si a esto se le suman ciertas metodologías y protocolos de seguridad como Secure Boot o autenticación, se mitigan gran parte de los ataques Shack y Hack. Todo esto en combinación con métodos que intentan mitigar ciertos ataques de laboratorio hacen que se tenga una solución robusta para la seguridad del dispositivo.

Arquitectura hardware El objetivo de TrustZone consiste en proporcionar una arquitectura hardware que permita que el dispositivo se ajuste a los requerimientos de seguridad de cada proyecto en lugar de aportar una solución basada en la idea de "una misma solución sirve para todo".

En primer lugar, esto se consigue creando un entorno de programación que garantice la confidencialidad y la integridad de todos los datos sensibles y los proteja de la gran mayoría de posibles ataques. Para ello, se realiza un particionado de la arquitectura hardware y software en dos entornos (o mundos como se llamarán a partir de ahora), un Mundo Seguro (MS) y un Mundo Normal (MN). En el Mundo Seguro se realizará todo lo que incumba tareas o datos sensibles y en el Mundo Normal todo lo demás. La lógica hardware presente en el bus AMBA3 AXI será la encargada de hacer que ningún recurso del MS sea accedido por ningún componente del MN, creando un perímetro de seguridad.

Otro aspecto clave de TrustZone es la extensión que se le ha dado a los procesadores ARM. Un único procesador podrá ejecutar código tanto del MS como del MN. Esto elimina la necesidad de tener un procesador que se encargue de las tareas del MS y otro para el MN. Esta característica hará al dispositivo más eficiente. Para conseguir esta separación en dos mundos, se han hecho varios cambios en algunos de los componentes de los dispositivos.

Bus AMBA3 AXI A cada uno de los buses del sistema se le ha añadido una nueva señal de control tanto en los canales de lectura como en los de escritura. Estos bits, añadidos a la especificación del protocolo del bus AMBA3 AXI, se denominan bits No-Seguros.

- AWPROT[1]: Escritura (baja es Segura y alta es No Segura).
- ARPROT[1]: Lectura (baja es Segura y alta es No Segura).

Todos los buses máster enviarán esta señal cuando se quiera realizar una acción de lectura o escritura, y la lógica de los buses o esclavos debe decodificarlas e interpretarlas correctamente para asegurar que esa separación de mundos no se viola.

Bus periférico AMBA3 APB Otra de las principales características de la arquitectura TrustZone consiste en la posibilidad de hacer seguros a los periféricos (controladores de interrupciones, temporizadores y dispositivos de Entrada/Salida). Esto permite una seguridad punto a punto en lugar de únicamente proporcionar una seguridad en el procesado de los datos.

Memoria La inclusión del bit No-Seguro al bus se puede ver como una partición en dos de la memoria. El bit No-Seguro se considera como un 33º bit en el espacio de direcciones físicas. Se tendrá por tanto un espacio de direcciones físicas de 32 bits para las transacciones seguras y un espacio equivalente de 32 bits para las no seguras. Se debe tener especial cuidado para asegurar la integridad y coherencia de los datos en todas las localizaciones en las que se almacenen.

Arquitectura del procesador Cada uno de los procesadores físicos provee al sistema de dos procesadores virtuales, uno considerado Seguro y otro No Seguro. Además, también proporciona un mecanismo para cambiar el contexto de ejecución de manera robusta entre Seguro y No Seguro, se le denomina Modo Monitor. Dependiendo de la identidad del procesador virtual desde el que se envíe una transacción, esta será etiquetada como segura o no segura haciendo uso del bit No-Seguro. Esto facilitará la integración de los procesadores virtuales en el sistema. El procesador virtual No Seguro

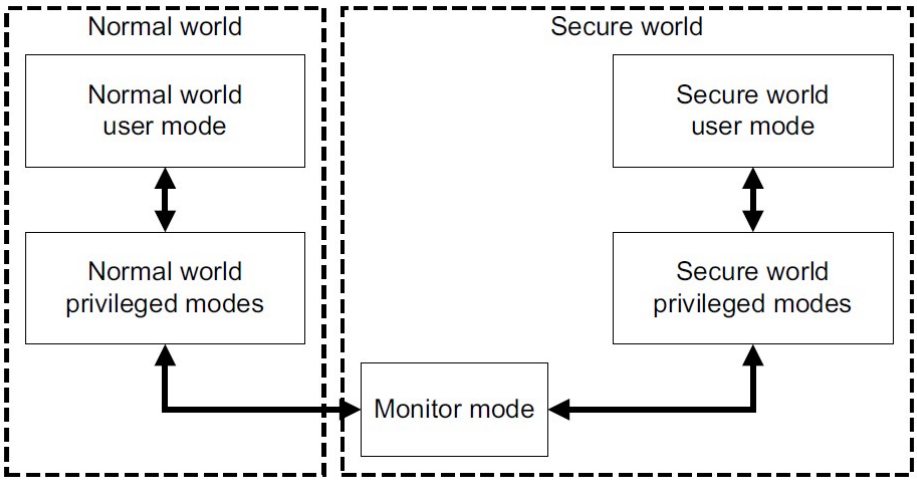

Figura 2.8: Estructura de un procesador físico TrustZone [28].

solamente podrá acceder a los recursos del sistema que sean no seguros. En cambio, el procesador virtual Seguro podrá acceder a todos los recursos del sistema.

Los mecanismos por los cuales el procesador físico entra en modo monitor son vistos como excepciones al software del modo monitor. La entrada al modo monitor puede estar provocada por el software que ejecuta una determinada instrucción (llamada Secure Monitor Call, SMC) o por mecanismos de excepción hardware (IRQ, FIQ, ...).

El mundo en el que el procesador se encuentra está marcado por el bit No-Seguro en el registro Secure Configuration Register (SCR) en CP15. Cuando se encuentra en modo monitor está siempre en el MS. Se ha mencionado que los espacios de memoria quedan divididos de manera que la memoria está separada para los dos mundos. Esta división que ocurre en la memoria fuera del procesador, debe ocurrir también en los componentes de memoria internos al procesador como es el sistema de memoria L1. El principal componente de la memoria L1 es el Memory Management Unit (MMU). El MMU es capaz de mapear el espacio de direcciones de memoria virtual que es visto por el software ejecutándose en el procesador con el espacio de direcciones físico existente fuera del procesador. La traducción se realiza haciendo uso de una tabla de traducción controlada por software. En un procesador con TrustZone, el hardware proporciona dos MMUs, una para cada procesador virtual. Esto permite que cada mundo posea su propia tabla de traducción de direcciones de manear totalmente independiente.

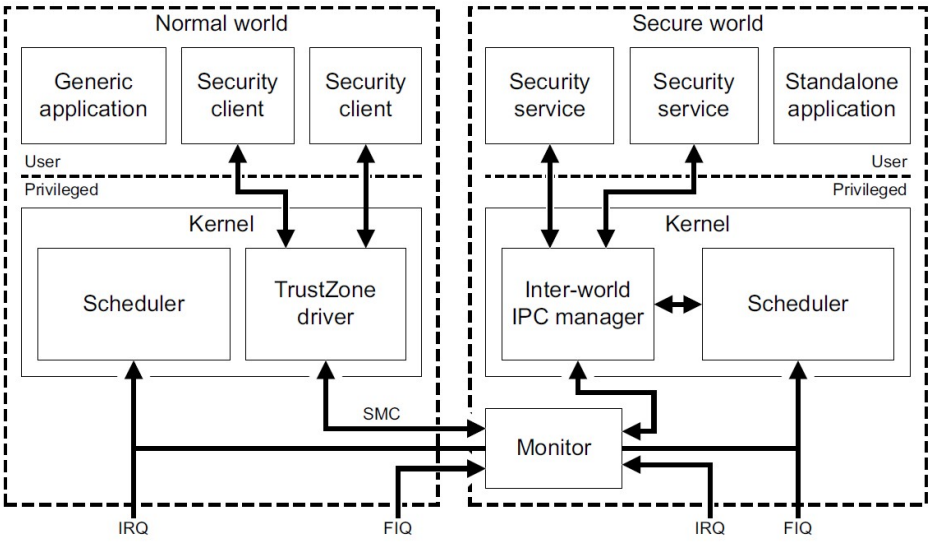

Figura 2.9: Esquema de un SO sobre el Mundo Seguro [28].

Arquitectura software La implementación de un Mundo Seguro en el dispositivo requiere de un software seguro que se ejecute y haga uso de los datos sensibles que se encuentren en este. Cada desarrollador puede crear un MS que se adecúe a las necesidades y requerimientos de su sistema.

Existen muchas posibles arquitecturas software que podrán ser desarrolladas en el Mundo Seguro de un procesador con TrustZone habilitado. La arquitectura más compleja que se puede alojar en un MS es un sistema operativo. La más sencilla es una librería de código alojada en el MS. Entre estos dos extremos existen un gran abanico de posibilidades intermedias.

Sistema operativo Un sistema operativo dedicado en el MS es un diseño complejo pero que abre un gran abanico de posibilidades. Puede simular la ejecución concurrente de varias aplicaciones del MS, descargar nuevas aplicaciones seguras y otro tipo de tareas seguras que serán totalmente independientes del MN. Cada procesador virtual tendrá un sistema operativo completo y se comportará como tal.

Librería de código Muchos casos de uso no justificarán la complejidad de un sistema operativo en el MS ya que sus requerimientos son mucho menores. Una sencilla librería de código que ejecute una tarea en el MS puede ser suficiente para muchos casos de uso. Esta librería se ejecutará mediante las pertinentes llamadas realizadas desde el sistema operativo del MN. El MS es por tanto un esclavo del MN ya que el MS no se podrá ejecutar

independientemente.

Arrancado de un sistema seguro Uno de los puntos más críticos de un sistema seguro es el arrancado. Muchos de los ataques se centran en la modificación del sistema cuando este se encuentra apagado. Por ejemplo, reemplazar la imagen del MS que se encuentra en la memoria flash del dispositivo con una imagen del MS modificada. Por tanto, para que el sistema no sea vulnerable se deberá comprobar la autenticidad del código en el tiempo de arranque. La estrategia que se sigue consiste en una cadena de confianza para el software del mundo seguro que partirá de un punto difícilmente manipulable. Esto se corresponderá con una determinada secuencia de arrancado.

Secuencia de arrancado Cuando el dispositivo se enciende, un procesador con TrustZone habilitado siempre comienza en el Mundo Seguro. Esto permite que se puedan realizar diversas comprobaciones de seguridad antes de que el Mundo Seguro arranque y tenga alguna oportunidad de modificar algún aspecto del sistema.

La figura XXXXXXXXXXXXx muestra una típica secuencia de arrancado. Tras encender el dispositivo se suele ejecutar un código que se encuentra en la ROM del dispositivo y que será el encargado de inicializar periféricos críticos del dispositivo. Este da paso a un código de arranque de segundo nivel que se encuentra en la memoria no-volátil externa del dispositivo. La secuencia continuará avanzando hasta que se haya completado el arranque del Mundo Seguro. Posteriormente se da paso al Mundo Normal y una vez este se haya completado se puede decir que el sistema se está ejecutando correctamente. Para que esta secuencia tenga sentido se deben añadir comprobaciones criptográficas en cada fase del arrancado seguro. Cada fase de la secuencia realizará una comprobación de la integridad de la siguiente fase a ejecutar. Se previene así que arranque cualquier tipo de código malicioso no autorizado o modificado.

Para esta tarea se utiliza un algoritmo de cifrado de clave asimétrica, RSA-PSS (Rivest, Shamir and Adleman – Probabilistic Signature Scheme). En este protocolo, un vendedor de confianza generará una firma del código que se quiere desplegar haciendo uso de su clave privada. Esta firma irá incluida en el binario del software. La clave pública del vendedor se almacenará en el dispositivo en alguno de sus registros dedicados a este propósito que no podrán ser alterados. Esta clave pública se utilizará para verificar que el binario proporcionado por el vendedor no ha sido modificado, por ello se necesita que esta clave pública no pueda ser modificada por una propiedad de un atacante. Se crea por tanto una cadena de confianza que

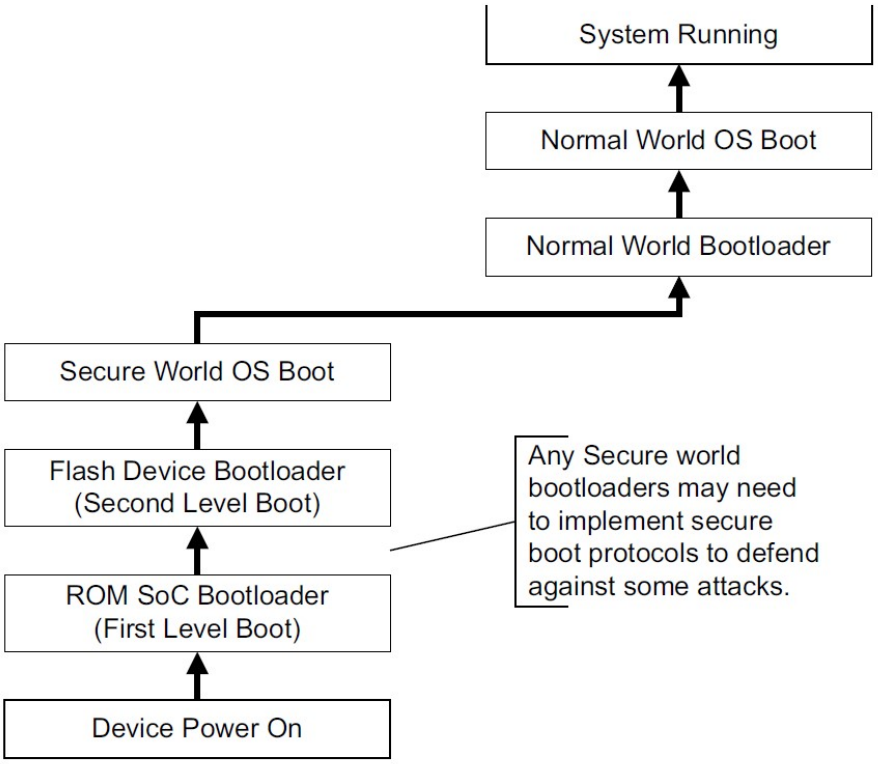

Figura 2.10: Secuencia de arrancado seguro [28].

nace en un componente del dispositivo en el que se confía. A partir de este se podrá autenticar cualquier otro componente o software antes de que sea ejecutado. Una vez que se ha ejecutado la autenticación de la primera fase de la secuencia, las próximas autenticaciones de la secuencia podrán ser realizadas con claves públicas diferentes. Solamente la autenticación de la primera fase debe realizarse con la clave que se encuentra en los registros One-Time-Programmable (OTP) del dispositivo.

2.3. Revisión de tecnologías Blockchain

En este apartado se describen (1) la historia y el conjunto de hitos más importantes que dan origen al proyecto Bitcoin y la tecnología blockchain y (2) las bases sobre las que se asienta el protocolo blockchain y su funcionamiento.

2.3.1. Origen e historia

La tecnología blockchain garantiza el correcto funcionamiento de los aspectos clave de las criptomonedas: la puesta en circulación de la moneda y el mecanismo de gasto que garantiza que una moneda no pueda ser gastada varias veces. Anteriores monedas digitales habían intentado resolver estos dos problemas sin éxito. El proyecto Bitcoin, cuya base tecnológica es blockchain, es la primera criptomoneda capaz de resolverlos de manera totalmente descentralizada.

Es vital para cualquier moneda digital que la emisión de la misma sea predecible y constante en el tiempo. La solución al problema del mecanismo para poner en circulación una moneda digital tiene su origen en el proyecto de Adam Back llamado HashCash. El proyecto tenía como objetivo combatir el correo spam haciendo uso de la técnica conocida como prueba de trabajo. La prueba de trabajo consiste en un problema matemático de resolución compleja pero cuya verificación es muy sencilla. El aspecto clave de la prueba de trabajo radica en que la dificultad del problema a resolver se puede ajustar. La regulación de la dificultad de la prueba de trabajo es el mecanismo por el cual se consigue ajustar la puesta en circulación. Las redes blockchain que hacen uso de la prueba de trabajo permiten que el usuario que antes resuelva uno de estos problemas emita (ponga en circulación) una determinada cantidad de nuevas monedas. El resto de usuarios de la red procederá a la sencilla verificación de la solución del problema antes de comenzar a buscar la solución a otro de estos problemas.

El segundo aspecto vital para una criptomoneda es conseguir un mecanismo de gasto que garantice que una moneda no pueda ser gastada varias veces. Para resolver el segundo problema se necesita que los usuarios de la

Figura 2.11: Precio Bitcoin [3].

red sean capaces de conocer el último estado consensuado de la blockchain. Gracias a esto, los usuarios de la red serán capaces de determinar sin un usuario de la red es el poseedor de cierta moneda o no. En otras palabras, los usuarios de la red pueden verificar si un usuario en concreto está realizando un doble gasto de una determinada moneda o no. El trabajo del investigador Nick Szabo sobre la tolerancia a fallos bizantinos se convierte en la pieza clave para que los usuarios de la red lleguen al consenso sobre el último estado global de la blockchain.

XXXXXXXXXXXXXTema 1 Alberto Toribio Intro a blockchain curso UNIR

La tecnología blockchain permite a los usuarios de la red tener un registro válido, inalterable y consensuado de todas las transacciones realizadas. Los bloques de la cadena contienen la información relativa a esas transacciones, es por esto que a menudo se denomina a la blockchain como libro contable distribuido.

El 3 de enero de 2009 Satoshi Nakamoto crea el primer bloque de la cadena de Bitcoin conocido como bloque génesis. Desde ese momento se han estado creando nuevos bloques de la cadena y nuevos bitcoins ininterrumpidamente. Sin embargo, la popularidad de Bitcoin no ha sido la misma desde entonces. Durante los primeros años sufrió varios altibajos tanto en popularidad como en precio. Como se observa en la figura XXXXXXXX, no fue hasta finales de 2017 cuando su popularidad y precio comenzaron a incrementarse de manera exponencial. Entre 2009 y 2017 surgen muchas otras criptomonedas y proyectos basados en Bitcoin y blockchain.

El proyecto con mayor repercusión que hizo ver las posibilidades que

ofrecía la tecnología blockchain más allá de Bitcoin fue Ethereum. El proyecto Ethereum, bajo el liderazgo de Vitalik Buterin, surge en 2014 con el objetivo de crear una nueva red blockchain. La red Ethereum, además de registrar todas las transacciones de su propia moneda digital (el Ether), registra pequeños programas informáticos llamados Smart Contracts. Los usuarios de la red, además de estar de acuerdo en las transacciones que se realizan en la red, deben estar de acuerdo en los despliegues de los Smart Contracts y de los resultados de la ejecución de los mismos. Con los resultados de la ejecución de los Smart Contracts ocurre exactamente igual que con las transacciones de monedas, si la mayoría de la red está de acuerdo (hay consenso) el resultado pasa automáticamente a ser válido.

El nombre de Smart Contract puede llevar a equivocación. Los Smart Contracts no extienden todas las capacidades o características de un contrato convencional. Son programas informáticos que permiten programar una serie de instrucciones máquina propias de su máquina virtual (Ethereum Virtual Machine, EVM) para interaccionar con los datos que residen en memoria. Por lo tanto, la clase de tareas que podrán realizar los Smart Contracts será muy limitada.

Uno de los principales usos de los Smart Contracts es la creación de tokens. Un token es una unidad de valor que una organización crea para gobernar su modelo de negocio y dar más poder a sus usuarios para interactuar con sus productos, al tiempo que facilita la distribución y reparto de beneficios entre todos sus accionistas [32]. El Smart Contract asociado a un determinado token será el encargado de programar el registro y la interacción que se tenga con el token en cuestión. En definitiva, se tendría un pequeño libro contable creado sobre un gran libro contable (la blockchain) [XXXXXtema 1 UNIR].

La tokenización de activos abre un gran abanico de posibilidades, siendo su mayor práctica las ICOs (Initial Coin Offering). Permite una nueva forma de financiación a nuevas empresas. Consiste en la preventa de su propia moneda digital o token en representación de los servicios que dicha empresa prestará en el futuro. En el 2017 la financiación obtenida por empresas que han realizado esta oferta inicial de moneda supera los 5000 millones de euros [XXXXXbuscar dato real].

Tras toda esta serie de puntos de inflexión que han marcado el avance de la tecnología blockchain, es necesario citar los principales problemas, aún por resolver, entre los cuales se encuentra la principal motivación de este trabajo.

La raíz de la mayoría de problemas está en la prueba de trabajo. La dificultad de los problemas de la prueba de trabajo se ajusta constantemente

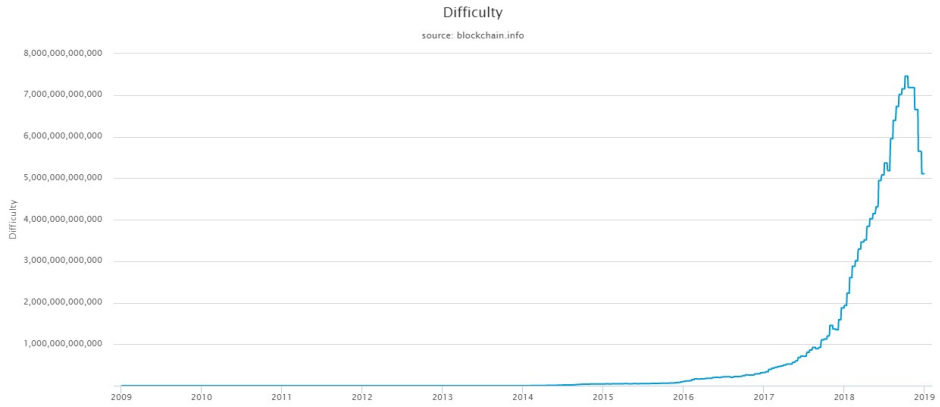

Figura 2.12: Dificultad de la prueba de trabajo Bitcoin [3].

para hacer que el tiempo que empleará el poder computacional de todos los usuarios de la red en resolver el problema sea constante. Este tiempo tiene que ser suficiente para que la solución al problema hallada por un usuario sea difundida a la mayoría de usuarios y se adquiera el consenso. Este tiempo dependerá de las características de la red blockchain en cuestión. Cada solución a un problema permite que se añada un nuevo bloque a la cadena que contendrá la información relacionada con las transacciones realizadas. Cada bloque tiene un tamaño máximo que dependerá de cada red blockchain. Se tendrá por tanto un tiempo de creación y un tamaño máximo prefijado. Esto se traduce en una tasa máxima de transacciones por segundo (tx/s) para cada blockchain. Notar que la dificultad de la prueba de trabajo se ajusta en base al poder computacional agregado de todos los usuarios para que el tiempo sea constante. Esto significa que no por añadir más poder computacional a la red se incrementará el número de transacciones por segundo. Se observa en la figura XXXXXXXx como la dificultad de la prueba de trabajo de Bitcoin ha ido aumentando en el tiempo de manera similar a su precio y popularidad. Esto se debe a que conforme crecía el interés por la criptomoneda se incrementaba el poder de cómputo proporcionado por los usuarios a la red.

En la figura XXXXXXXX se observan el número de transacciones confirmadas por día entre 2017 y 2019, años en los que se observa el mayor incremento en la dificultad de la prueba de trabajo. Este incremento de poder de cómputo (traducido en incremento de la dificultad de la prueba de trabajo), no se traduce en un claro incremento en el número de transacciones por día.

Si un sistema no aumenta su rendimiento cuando se le añaden más recursos al mismo se dice de este que no es escalable. Es por esto que las

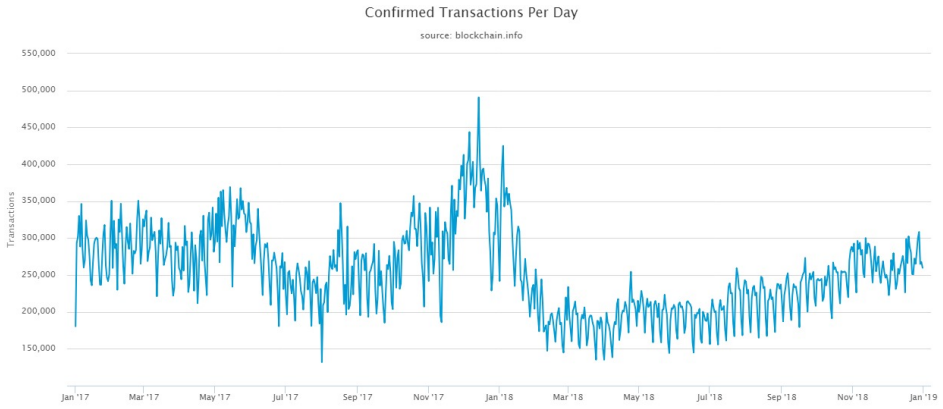

Figura 2.13: Número de transacciones Bitcoin confirmadas por día [3].

redes blockchain basadas en prueba de trabajo son ineficientes por diseño. En base a la definición que se tiene, no se podrá conseguir que sea escalable como tal. Sin embargo, se podrán realizar cambios en el protocolo para hacer que sea lo más eficiente posible. Estos cambios en el protocolo que hagan que se consiga la máxima eficiencia posible están aún por descubrir y diseñar.

La falta de escalabilidad da paso a otro de los principales problemas, el desperdicio de energía eléctrica. La reciente popularidad de las criptomonedas y las redes blockchain ha atraído el interés de muchas personas. Cada vez son más los usuarios que suman su poder computacional a la red con el objetivo de resolver los problemas de la prueba de trabajo. Sin embargo, el protocolo define que el tiempo empleado por el poder computacional agregado de todos los usuarios sea aproximadamente el mismo siempre. En otras palabras, si se aumenta el poder computacional de la red, aumentará también la dificultad de la prueba de trabajo para que entre todos tarden ese tiempo constante en resolverla. Este aumento de la dificultad se traducirá en un aumento de la energía eléctrica empleada en resolver la prueba de trabajo. El resultado siempre será la creación de un nuevo bloque de la cadena cada cierto intervalo de tiempo, la consecuencia directa que cuánto más poder computacional tenga la red, más energía eléctrica se empleará en el proceso. Existe mucha controversia acerca de si en realidad se trata de un desperdicio de energía eléctrica o no. Sin embargo, este debate no es el tema de este trabajo. Cuando un usuario encuentra solución a la prueba de trabajo, este obtiene la capacidad de crear cierta cantidad de nuevas monedas que se autoasignará. Esto ha provocado la creación de grupos que aúnan todo su poder computacional en un mismo usuario para así aumentar la posibilidad de encontrar la solución a la prueba de trabajo recibir estas nuevas monedas creadas que posteriormente repartirán entre todos. Surge así un nuevo problema que atenta contra la descentralización de la red. En redes como la de

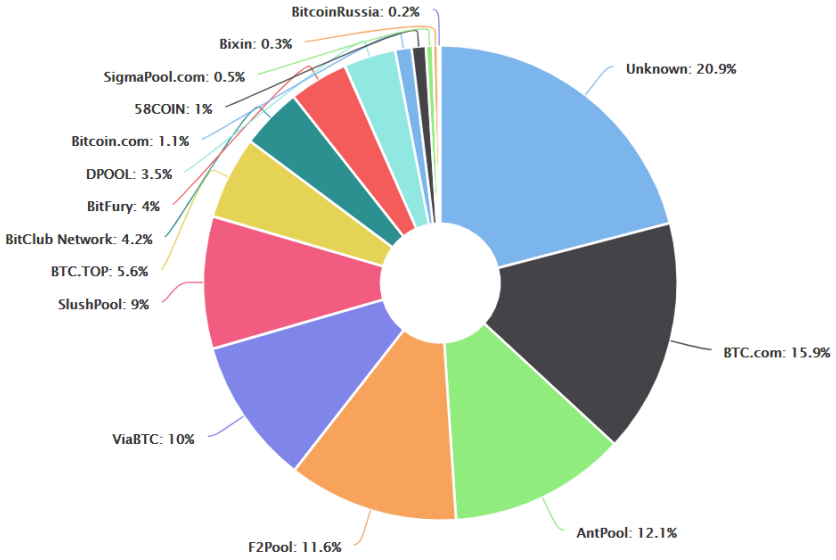

Figura 2.14: Porcentaje de pruebas de trabajo resueltas por cada grupo [3].

Bitcoin, casi la totalidad de los nuevos bloques son incluidos a la cadena por tres de estos grupos como se observa en la figura XXXXXXXXXXXXXX. Se podría decir que estos tres grupos gobiernan la blockchain de Bitcoin. A esto se le suma que, si uno de estos grupos adquiere al menos el 51 % del poder computacional de la red, este podrá incluir las transacciones (válidas o inválidas) que desee a su antojo. Esto se conoce como el ataque del 51 % y es posible ya que este grupo haría que la mayoría de la red esté en consenso. En la figura XXXXXXXXXXXXXX se observa el porcentaje de éxito de cada uno de estos grupos durante el día 1 de enero de 2019. Notar que esta distribución se puede considerar constante durante el último año.

La gran cantidad de inconvenientes aún por solventar o mejorar motivan este trabajo. En el siguiente apartado se definirá más concretamente el problema que se pretende atajar en el trabajo y los objetivos del mismo.

2.3.2. Funcionamiento

En el apartado de Origen e Historia se ha dado una visión muy general de cómo surgió blockchain y de sus hitos más importantes. La explicación de las redes blockchain ha sido tan simple como ha sido posible para entender la motivación y los objetivos del proyecto. Sin embargo, es necesario conocer con más detalle el funcionamiento y las posibilidades que ofrecen los

distintos tipos de redes blockchain para la correcta comprensión del trabajo. Notar que, pese a que la explicación se realizará en base a la blockchain de Bitcoin, la explicación es extensible a la gran mayoría de redes blockchain.

2.3.2.1. Criptografía

La criptografía pese a ser prácticamente transparente para el usuario, juega un papel fundamental en la actualidad. Acciones tan cotidianas como enviar un correo electrónico o pagar con tarjeta de crédito funcionan gracias a la criptografía. La criptografía aporta seguridad a estos procedimientos. ¿Por qué se confía en la criptografía? La criptografía se basa en una ciencia exacta, como son las matemáticas, y en algoritmos muy estudiados por la comunidad científica. Varios de estos algoritmos de la criptografía son usados en la tecnología blockchain para aportar seguridad. Son esenciales para su correcto funcionamiento.

Las funciones Hash La traducción al castellano de blockchain es cadena de bloques. En este apartado se explica la metodología que se sigue para que los bloques de información queden encadenados y cuál es el objetivo de esto. Una función hash es una función unidireccional. Las funciones unidireccionales se definen como:

XXXXXXx f:X – Y,f(x)=y
La principal característica de estas funciones es que f(x) es fácil de calcular, pero calcular fXXXX(-1) (y) es computacionalmente irrealizable. Apoyadas en esta característica, las funciones hash se aplican sobre mensajes M para proporcionar un resumen del mismo denominado m. El resumen m tendrá una longitud fija independiente de la longitud del mensaje M. Existen numerosas funciones hash. Sin embargo, para que una función hash se considere segura y pueda ser utilizada en criptografía debe cumplir una serie de requisitos:

1. 1. Dependencia de bits: se corresponde con la propiedad de dispersión de las funciones hash. El resumen debe depender de todos los bits del mensaje. Esto significa que, si se cambia un bit del mensaje, cambian de media la mitad de los bits del resumen.

2. 2. Resistencia a la preimagen: dado un resumen m, debe ser computacionalmente difícil obtener M de modo que H(M) = m.

3. 3. Resistencia a la segunda preimagen: dado un mensaje M, debe ser computacionalmente difícil encontrar otro mensaje N cuyos resúmenes coincidan, es decir H(M) = H(N).

4. 4. Resistencia a colisiones: debe ser computacionalmente difícil encontrar una colisión. Una colisión consiste en dos mensajes M y N cuyos resúmenes coincidan.

En definitiva, las funciones hash permiten obtener un identificador único de un mensaje. Identificador a partir del cual no se podrá obtener el mensaje original y que garantiza la integridad del mismo. Este aspecto es lo que hace a las funciones hash indispensables para blockchain.

Aplicación de las funciones Hash a Blockchain Una cadena de bloques se origina a partir de un primer bloque, el bloque génesis. Se pretende que todos los bloques que se vayan añadiendo a la cadena estén enlazados de alguna manera inequívoca con todos los bloques previos hasta llegar al génesis. Esto se consigue con las funciones hash siguiendo la siguiente metodología desde el bloque génesis:

1. 1. Haciendo uso de una función hash, se obtiene el resumen o hash del bloque génesis. El hash será un código de longitud fija que identifica inequívocamente al bloque génesis.

2. 2. Para crear el siguiente bloque de la cadena, se incluye en este un campo que contendrá el hash del bloque anterior.

3. 3. Una vez finalizada la creación del nuevo bloque, se le calcula el hash al nuevo bloque y se vuelve al paso 2.

Con este procedimiento se consigue que el hash de un bloque N de la cadena identifique inequívocamente a toda la cadena de bloques desde el bloque génesis hasta el bloque N. Se garantiza que toda la cadena de bloques permanece íntegra y que si alguno de los bloques sufre algún cambio se alterarán los hashes de todos los bloques posteriores. El resultado partiendo desde el bloque génesis se observa en la figura XXXXXXXXXXXXxxx.

Criptografía de clave asimétrica En 1976 Diffie y Hellman proponen un nuevo método para la distribución de claves. El propósito de este método era permitir a dos extremos de una comunicación establecer una clave simétrica sin que estos hayan tenido un contacto previo o usen un canal seguro. Esta propuesta daba solución al intercambio de claves simétricas, pero no permitía el cifrado. Sin embargo, este método dio origen un año después al primer algoritmo de cifrado de clave asimétrica. En 1977 nace el algoritmo RSA de la mano de Rivest, Shamir y Adleman. Este es uno de los algoritmos más utilizados incluso a día de hoy. RSA permite no sólo el cifrado sino también la firma digital.

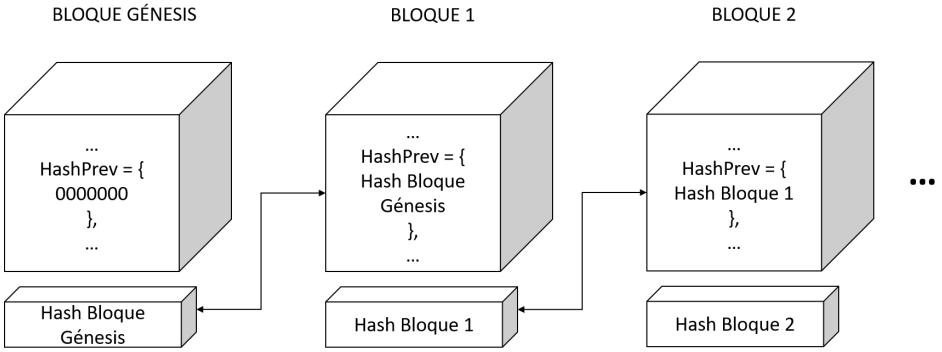

Figura 2.15: Estructura de hashes de Blockchain. XXX

La principal innovación que introduce este tipo de criptografía es el uso de un par de claves. Se tendrá una clave pública y una clave privada. Como su propio nombre indica, la clave pública será conocida por todos, en cambio la clave privada sólo la debe conocer su propietario. De esta manera deja de ser necesario el intercambio de una clave simétrica a través de un canal seguro. La criptografía asimétrica se apoya en tres tipos de problemas matemáticos:

1. Problema de la factorización de enteros
2. Logaritmo discreto
3. Curvas elípticas

Al igual que ocurría con las funciones hash, las funciones matemáticas escogidas para estos problemas tienen como principal característica el fácil cálculo en un sentido, pero muy difícil computacionalmente en el sentido inverso. De nuevo se trata de funciones unidireccionales.

El cifrado de un mensaje mediante criptografía de clave simétrica sigue el siguiente esquema:

1. El emisor obtiene la clave pública del destinatario por cualquier canal (puede ser inseguro).
2. El emisor cifra el mensaje con la clave pública del destinatario.
3. El emisor envía el mensaje cifrado a través de cualquier canal ya que solamente el poseedor de la clave privada asociada a la clave pública con la que se cifró el mensaje podrá descifrarlo. En este caso el destinatario.
4. El destinatario recibe el mensaje cifrado y haciendo uso de su clave privada obtiene el mensaje en texto plano.

Este esquema explica un simple intercambio de un mensaje cifrado. Sin embargo, la criptografía de clave asimétrica abre un abanico de posibilidades a protocolos más complicados que ofrezcan distintas cosas. Un claro ejemplo es la firma digital. La tecnología blockchain también hace uso de este tipo de cifrado.

Los principales aportes que hacen al cifrado asimétrico tan interesante para blockchain son:

1. No repudio. Si un usuario firma un mensaje con su clave privada, no podrá negar haberlo hecho.

2. Identificación. Ambos extremos de una comunicación quedan identificados.

Aplicación de la criptografía a Blockchain Este apartado describe el uso que la tecnología blockchain hace de la criptografía de clave asimétrica. Una red blockchain es una red P2P distribuida, por lo tanto, ninguno de los usuarios es a priori conocido o fiable. Dentro de esta red la identidad de los usuarios es pseudoanónima. No se conoce la identidad real de los usuarios, pero tampoco es necesario. Solamente es necesario un identificador único que los identifique dentro de la red. Cada usuario tiene un par de claves asimétricas (pública y privada). Este identificador único se corresponde con la clave pública del usuario (existe un proceso que transforma la clave pública en una dirección, pero no es relevante).

La mejor forma de observar el uso que se hace de este par de claves es describir los pasos que se siguen para realizar una transacción en la blockchain de Bitcoin.

1. El receptor envía su clave pública al emisor de la transacción.

2. El emisor incluye en el campo destinatario de la transacción la clave pública del destinatario.

3. La transacción, entre otros datos, incluye la cantidad de bitcoins a transferir.

4. El emisor cifra la transacción con su clave privada. En este paso el emisor está demostrando que posee una cierta cantidad de bitcoins y acepta que ahora pasan a ser propiedad del destinatario.

5. La transacción se transmitirá a todos los usuarios de la red que la validarán e incluirán en la blockchain.

El paso 4 resulta crucial ya que de él se obtiene la identificación y el no repudio de la transacción. Resulta evidente que la criptografía de clave asimétrica es idónea y totalmente necesaria para el correcto funcionamiento de cualquier red blockchain.

2.3.2.2. Teoría de juegos

La teoría de juegos es una rama de las matemáticas aplicada que analiza la toma de decisiones en situaciones en las que la decisión no sólo se basa en el punto de vista de un jugador, sino teniendo en cuenta las decisiones tomadas por el resto de jugadores. Se entiende por juego una situación en la que un jugador podrá obtener recompensas o perjuicios según la decisión que hayan tomado él mismo y el resto de jugadores.

El concepto relacionado con la teoría de juegos más importante para blockchain es el equilibrio de Nash. Este concepto, creado por John Nash, afirma que, si todos los jugadores siguen una estrategia óptima, ningún jugador saldrá beneficiado si cambia individualmente su estrategia. El equilibrio de Nash está directamente relacionado con la expresión "Si la mayoría de los usuarios de la red actúa correctamente la blockchain funcionará correctamente". Ciertos usuarios de la red podrían intentar hacer trampa incluyendo bloques a la cadena sin resolver la prueba de trabajo, incluir transacciones inválidas, etc. Sin embargo, blockchain está diseñada para que se mantenga el equilibrio de Nash. Si se incluye un bloque válido encima de uno inválido, este pasa automáticamente a ser inválido también. Por lo tanto, la estrategia que les interesa a los usuarios es actuar de manera correcta y elegir la situación más estable. Se mantendrá el equilibrio de Nash mientras la mayoría de los usuarios actúen de manera honesta. Si usuarios con más de la mitad del poder computacional de la red deciden actuar de manera deshonesta, el equilibrio de Nash pasará a localizarse del lado de los usuarios deshonestos y la blockchain pasará a estar corrompida. Esta es la explicación del famoso ataque del 51 %.

2.3.3. Aspectos clave del protocolo Blockchain

Una vez se tiene la base matemática de la base de las tecnologías que combina blockchain se explican las partes más relevantes del protocolo.

2.3.3.1. Concepto de minado

Hasta este apartado se había descrito la red blockchain como una red P2P formada por usuarios. Se utiliza el termino usuario por simplicidad. En este apartado se explica en qué consiste la minería de bloques en blockchain y quienes son los que la realizan.

En realidad, la red blockchain la componen una serie de nodos denominados mineros. Los usuarios simplemente son los encargados de enviar sus propias transacciones a la red cuando lo deseen. La tarea de los mineros consiste en minar bloques que contengan transacciones con el objetivo de obtener una recompensa. Minar un bloque es como se denomina al acto de resolver la prueba de trabajo descrita de manera simplificada en el APARTADO XXXXXXXXXXXXXxxX.

Prueba de trabajo (Proof of Work(PoW)) En el apartado XXXXXXXXxx se describe la prueba de trabajo como un problema de solución muy compleja, pero de verificación muy sencilla. Existen multitud de problemas matemáticos que se podrían utilizar para la prueba de trabajo de blockchain. El problema elegido para blockchain es el cálculo de un resumen hash con una característica determinada. El hash que sirva de solución a la prueba de trabajo de un bloque debe tener una determinada cantidad de ceros ('000...') al principio. La cantidad de ceros estará determinada por la dificultad, que a su vez está marcada por la cantidad de poder computacional total de la red. El resumen hash se calcula al bloque que se pretende incluir en cadena de bloques. Por lo tanto, los pasos que deberá seguir un minero que pretende resolver la prueba de trabajo y minar un nuevo bloque son los siguientes:

1. Calcular la dificultad del siguiente bloque de la cadena. Esta dificultad se calcula en base a una expresión que depende del tiempo que se ha tardado en minar los últimos bloques de la cadena. La dificultad determina el número de ceros que debe tener el hash del siguiente bloque a incluir en la cadena.

2. Construir un bloque válido que pretenda incluir en la cadena.

3. Dentro del bloque existe un campo denominado nonce que será variable y que no aporta ninguna información. El minero deberá ir cambiando el campo nonce hasta que obtenga un hash del bloque que tenga el número de ceros que satisfaga la prueba de trabajo.

4. Dar un valor al campo nonce.

5. Calcular el hash del bloque con el nonce incluido en el punto 4.

6. Si el hash obtenido no satisface la prueba de trabajo, volver al punto 4. En caso de que el hash satisfaga la prueba de trabajo, se dice que se ha minado el bloque. El minero puede proceder a difundir este bloque minado para que los demás lo validen e incluyan a la cadena de bloques. El hecho de incluir el bloque en la cadena significa que cuando el resto de mineros vuelvan a realizar este algoritmo, en el paso 2 incluyan el hash del bloque recién minado en el campo hash previo. Así este permanecerá en la cadena inalterable para siempre.

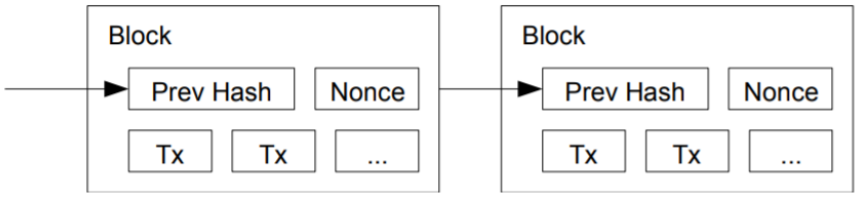

Figura 2.16: Estructura de bloques de Blockchain [33]

Este es el algoritmo que deberá seguir todo minero para minar un nuevo bloque. Notar que es en los puntos 4, 5 y 6 donde se emplea la gran parte del poder de cómputo para el minado.

Creación de un bloque válido En el algoritmo que debe seguir un minero para minar un nuevo bloque se observa que el segundo paso es crear un nuevo bloque válido a incluir en la cadena. Para entender en qué consiste este paso en detalle es necesario saber de qué está compuesto un bloque.

Un bloque está compuesto por un campo que contiene las últimas transacciones realizadas, otro campo que contendrá el hash del bloque anterior y el campo nonce descrito en el apartado anteriorXXXXXXXXXXXXXXXXXXXX y observado en la figura XXXXXXXXXx. Para que un bloque sea válido, todos los datos que este contenga deben ser válidos. En especial es necesario validar las transacciones que se incluyan, ya que si nuestro bloque es inválido el resto de la cadena lo rechazará para mantener el equilibrio de Nash. La validación de las transacciones consiste básicamente en dos pasos:

1. Validar la clave con la que se ha cifrado la transacción.

2. Comprobar que esa clave privada realmente posee la cantidad de moneda digital que pretende transferir.

2.3.3.2. Validez de la cadena de bloques

En ninguno de los procesos seguidos en las redes blockchain se puede olvidar que se trata de una red distribuida P2P compuesta por nodos desconocidos. No se tiene la certeza de que los nodos con los que se interactúe estén actuando de manera honesta. Puede ocurrir que un nuevo nodo quiera entrar a formar parte de la red blockchain. Este nuevo nodo desconoce totalmente lo sucedido en la cadena de bloques anteriormente así que debe pedirla a varios de los nodos que ya forman parte de la red. Si cada uno de los usuarios a los que solicita la cadena de bloques le proporciona una cade-

na de bloques distinta, este nuevo nodo deberá saber cuál elegir por sí mismo.

De entre todas las posibles cadenas, la cadena válida (y en la que se encuentra el equilibrio de Nash) se determinará llevando a cabo el siguiente procedimiento:

1. De entre todas las cadenas de bloques, se debe quedar sólo con las cadenas bien formadas. Una cadena está bien formada si todos los bloques que esta contiene son válidos (incluyendo sus transacciones) y todos los bloques están enlazados correctamente mediante hashes.

2. De entre todas las cadenas bien formadas, la cadena válida será la que más carga computacional contenga. Cada bloque añade carga computacional a una cadena de bloques, por lo tanto, la cadena válida será la más larga (en número de bloques).

3. Se puede dar el caso de que el paso 2 nos de como resultado varias cadenas posibles. En este caso el nuevo nodo deberá elegir al azar entre todas las posibilidades y esperar a que en el futuro se llegue al consenso con los dos primeros pasos del procedimiento.

La situación planteada no solamente se puede dar cuando un nuevo nodo quiere entrar a formar parte de la red. Otro caso en el que se puede dar es cuando varios mineros minan un bloque en un tiempo similar y cada uno transmite su bloque minado desde un punto distinto de la red. Un nodo intermedio puede recibir varias cadenas de bloques distintas y deberá elegir la cadena válida en base al procedimiento planteado. Cuando se mina un bloque simultáneamente en dos puntos de la red y ambas cadenas llegan a un nodo intermedio se dice que se ha producido un fork. Un fork no es más que una cadena de bloques cuyo bloque sucesor pueden ser varios bloques durante un pequeño espacio de tiempo hasta que se consigue el consenso.

2.3.3.3. Incentivo

En el algoritmo descrito en la sección XXXXXXXXXXXXXXXx se menciona que los últimos 3 pasos son los que consumen gran cantidad de poder computacional y por ende de energía eléctrica. Estos mineros están trabajando para el correcto funcionamiento de la red, y por supuesto no lo hacen de manera altruista. Invierten dinero en hardware específico para minado y en energía eléctrica a cambio de una recompensa.

En el apartado XXXXXXXXXxx se describe la estructura de un bloque. Uno de los campos es el de transacciones. En este campo aparecen todas las transacciones que se han realizado recientemente y que el minero ha validado. Sin embargo, el protocolo permite al minero que consigue minar un

bloque incluir una transacción especial en ese campo de transacciones. Esta transacción genera una cantidad de moneda digital (determinada por el protocolo) que se autoasigna el minero. Esta es la recompensa que obtiene el minero y que sirve de incentivo a todos los mineros de la red para intentar minar el siguiente bloque.

Además de esta recompensa, es posible que las transacciones añadan un campo con una pequeña comisión que también se lleva el minero. Se hace uso de esta comisión en las ocasiones en las que la red está colapsada, haciendo que los mineros elijan estas transacciones antes que otras sin comisión. Notar que el bloque tiene un tamaño máximo y no se pueden incluir todas las transacciones que se deseen.

2.3.3.4. Otros algoritmos de consenso

Uno de los factores más importantes y que no se puede descuidar de la blockchain es el consenso. La historia que se escribe en la cadena de bloques debe ser común para todos los nodos. La prueba de trabajo es un algoritmo de consenso que consigue que en la cadena de bloques se vayan incluyendo transacciones en un orden lógico y sin que se produzca doble gasto con ninguna de ellas. Puede ocurrir que las transacciones se inserten en la cadena de bloques en un orden temporal distinto al orden en el que se realizaron, pero esto no importa. Lo único que importa para los mineros es que sean transacciones válidas en relación a los bloques anteriores, y todo lo que ocurra que no quede registrado en la cadena carecerá de validez para ellos.

La mayoría de redes blockchain utilizan como algoritmo de consenso la prueba de trabajo. Sin embargo, este algoritmo de consenso no es el único que existe. En la sección XXXXXXXXXXXXXXXXXx se explica por qué la prueba de trabajo se ha convertido en un impedimento para la escalabilidad. La comunidad blockchain se ha dado cuenta de esto y se han desarrollado otra serie de algoritmos de consenso, algunos de los cuales se encuentran en producción, y la mayoría en desarrollo.

Prueba de participación (Proof of Stake (PoS)) Existe un enfoque que no se ha visto sobre la prueba de trabajo y por qué la red de Bitcoin no ha sido aún corrompida. Si un minero decide realizar un ataque de más del 51 % de poder de cómputo a la red, estaría consiguiendo minar todos los bloques (llevándose todas las recompensas) e insertando todas las transacciones (válidas o inválidas) que desee. Sin embargo, si esto ocurre estaría consiguiendo apropiarse de una cantidad de moneda digital que carecerá de valor, ya que toda la red puede observar este ataque y ver que la cadena de bloques ha sido corrompida. Los mineros que se llevan recompensas son los primeros interesados en que la red funcione honestamente y que se siga man-

teniendo el equilibrio de Nash para que la moneda digital siga teniendo valor.

La idea de la prueba de participación es en el fondo similar. Los nodos que más cantidad de moneda digital poseen serán los más interesados en que la red siga funcionando de manera honesta y la moneda digital asociada a esta mantenga su valor. Por esto, estos nodos serán los encargados de hacer que en la cadena de bloques se inserten solamente transacciones válidas.

El protocolo se basa en la cantidad de monedas que tenga cada participante que quiera validar nuevos bloques. En lugar de minar un bloque se utiliza validar un bloque para referirse al acto de crear un nuevo bloque aceptado por el resto de la red. Cada nodo que quiera entrar a formar parte del grupo de nodos validadores debe demostrar la cantidad de monedas que posee. Para ello, sus fondos quedarán "bloqueados" tanto tiempo como permanezcan al grupo de validadores. De entre todos los participantes que pertenezcan al grupo de validadores, un participante tendrá una probabilidad de ser el validador del siguiente bloque proporcional a los fondos que ha dejado "bloqueados". Es decir, se elegirá al validador del siguiente bloque aleatoriamente con una probabilidad igual a la prueba de participación que ese participante haya demostrado tener. Es por esto por lo que se dice que los que más cantidad de monedas tengan serán los encargados de velar por el correcto funcionamiento de la cadena de bloques. Este algoritmo está aún en fase de desarrollo, principalmente por una red blockchain llamada Ethereum. Esta red blockchain se explica en el apartado XXXXXXXXXXXXXXXXXxxx.

Prueba de tiempo expirado (Proof of Elapsed Time (PoET)) El objetivo final de los algoritmos de consenso es elegir de manera distribuida a uno de entre todos los nodos para que se encargue de crear el siguiente bloque de la cadena. En la prueba de trabajo se hace demostrando que se ha empleado cierto poder de cómputo y en la prueba de participación demostrando que se posee cierta cantidad de moneda. En este algoritmo se hace demostrando que tu procesador ha estado cierto tiempo aleatorio (sobre el que el usuario no tiene el control) durmiendo. Este algoritmo está aún en fase de desarrollo. Fue ideado por Intel aprovechando una de las características de algunos de sus procesadores llamada Trusted Execution Enviroment. El usuario no podrá interactuar con el software que se ejecute en ese ambiente de ejecución.

Estos son solo algunos de los algoritmos de consenso más llamativos del panorama actual. Existen otros muchos también en fase de desarrollo. El único que está demostrando conseguir el consenso en una red blockchain pública (DIFERENCIA ENTRE PUBLICAS Y PRIVADAS VER APTDO

XXXXXXXXXX) de manera segura es la prueba de trabajo, pese a los inconvenientes que suscita.

2.3.3.5. Redes blockchain públicas y permisionadas

Hasta el momento no se ha hecho mención a la diferencia entre las redes blockchain públicas y las permisionadas. Las redes objetivo del presente trabajo son las redes públicas como Bitcoin o Ethereum, pero conviene conocer en qué consisten las permisionadas ya que el trabajo puede tener aplicación también en estas. En una red blockchain pública, cualquiera puede empezar a formar parte de la red sin más. No será tan sencillo en las permisionadas.

Tras la aparición en 2014 de Ethereum y sus Smart Contracts, varias empresas ven cómo blockchain podría aportar valor a sus empresas facilitando el manejo de información. Hasta ese momento cuando varias empresas colaboraban entre sí en algún proceso, cada una gestionaba por separado la información relacionada con el mismo. Procesos de conciliación eran necesarios para cerciorarse de que la información que cada empresa tenía estaba en sintonía. Ciertas empresas consideran que esto puede cambiar con blockchain. Deciden crear consorcios con el objetivo de gestionar una red blockchain propia en la que almacenar de forma compartida esa información común y así agilizar muchos procesos. Estas nuevas redes blockchain se denominan permisionadas (o privadas en algunos casos). El término permisionada hace referencia a que, a diferencia de las redes públicas, es necesario que el resto de la red identifique a un nodo y lo autorice a participar antes de que este forme parte de la red. Esto se debe a que la información que se maneja en esta red puede ser sensible o confidencial para unas cuantas empresas. En estas redes blockchain se abre la posibilidad de que ciertas transacciones de información sean privadas y sólo unos cuantos puedan verlas. Otro de los aspectos claves de este tipo de redes blockchain es el rendimiento. Se pueden conseguir tasas de transacciones por segundo mucho mayores que las que se consiguen en las públicas. Esto se consigue a costa de cambios en el protocolo que hacen a esta red menos segura, ya que sólo ciertos nodos serán los encargados de incluir bloques a la cadena. Al tratarse de nodos identificados y autorizados esta disminución en la seguridad es tolerable. La elección entre pública o permisionada dependerá de la aplicación que se le quiera dar, las prestaciones que se necesiten y del ambiente en el que se ejecute (público o privado).

Dos claros ejemplos de redes blockchain permisionadas son Hyperledger Fabric de IBM y un consorcio de empresas españolas llamado Alastsria.

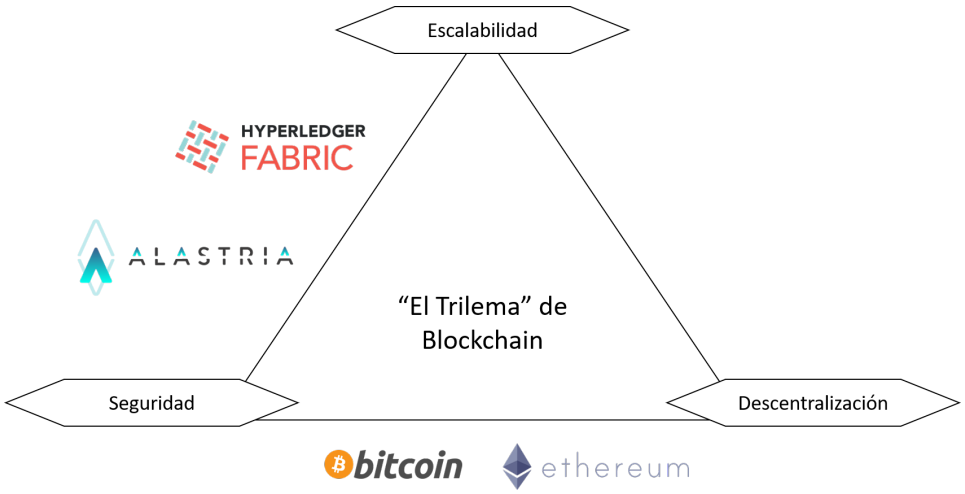

Figura 2.17: El Trilema de Blockchain. XXX

2.4. El problema de escalabilidad de Blockchain

Desde que surge el protocolo blockchain para Bitcoin, continuamente han estado apareciendo nuevos tipos de protocolos blockchain y de alternativas a blockchain. La gran mayoría de novedades se centran en mejorar la escalabilidad y la eficiencia (en términos de transacciones por segundo soportadas). Esto deja claro que la mayoría de usos a los que se desea aplicar requieren protocolos más eficientes.

2.4.1. El Trilema

En el apartado XXXXXXX sobre redes blockchain permisionadas se ha visto como estas consiguen más rendimiento (o escalabilidad) a costa de reducir la seguridad o la descentralización. Esto da paso a lo que en la comunidad blockchain se conoce como El Trilema.

Las redes blockchain tienes 3 aspectos fundamentales a tener en cuenta: seguridad, descentralización y escalabilidad. El problema se denomina trilema porque los protocolos blockchain desarrollados hasta el momento sólo son capaces de ofrecer 2 de estos aspectos poniendo en compromiso al restante. Redes blockchain públicas como la de Bitcoin o Ethereum ofrecen seguridad y descentralización a costa de no ser sistemas escalables. Redes blockchain permisionadas como Hyperledger Fabric o Alastria ofrecen escalabilidad y seguridad a costa de la descentralización (ya que sólo ciertos nodos elegidos incluyen bloques a la cadena). Actualmente no existe ningún protocolo blockchain que no comprometa alguno de estos aspectos clave. Conseguir solucionar el problema de la descentralización en las redes permisionadas es

imposible por definición, pese a esto existen muchos casos de uso en las que las redes blockchain permisionadas aportan valor. La otra opción es atajar el problema de la escalabilidad en redes públicas para intentar solucionarlo o al menos mejorar el rendimiento lo máximo posible. Esta última opción consiste en mejorar el algoritmo PoW o crear nuevos algoritmos de consenso.

2.4.2. Alternativas a Blockchain

Todos los esfuerzos no se ponen en intentar mejorar blockchain, al mismo tiempo surgen tecnologías alternativas a blockchain como Tangle o Hashgraph.

2.4.2.1. Tangle

Se denomina Tangle a la estructura de datos que soporta a IOTA [22]. IOTA es un proyecto que crea una criptomoneda con especial propósito para dispositivos IoT. La estructura de datos de Tangle consiste en un grafo de transacciones al que se irán añadiendo nuevas transacciones. Cada vez que se quiere añadir una nueva transacción es necesario validar dos transacciones anteriores. Necesita por tanto un gran volumen de transacciones para su correcto funcionamiento y seguridad, es por esto por lo que se pretende que su principal caso de uso sean los dispositivos IoT.

Sin embargo, el proyecto IOTA está en desarrollo ya que hay dudas al respecto de su seguridad. Para que el protocolo sea seguro necesita un alto volumen de transacciones, las cuales ahora mismo no existen, así que se hace uso de un ente centralizado que valida transacciones regularmente. Otra de las dudas surge cuando en su *whitepaper* asume que no existe entidad que pueda generar tal número de transacciones como para llevar a cabo un ataque en el que valida sus propias transacciones a su antojo. Dichos problemas estaban presentes al momento de iniciar el presente trabajo. Detrás del proyecto IOTA existe un gran número de desarrolladores que trabajan para solucionar dichos inconvenientes y llevar a cabo el proyecto con éxito.

2.4.2.2. Hashgraph

La tecnología Hashgraph [12] parece durante un tiempo que puede dejar obsoleta a blockchain. Una alternativa de consenso que utiliza un *gossip protocol* promete soportar unas 250 mil transacciones por segundo. Sin emgargo, la comunidad inmediatamente se percata de que Hashgraph solamente tiene cabida en una red permisionada y que si se intentase desplegar en una red pública tendría los mismos problemas que blockchain. Por esto se dice que Hashgraph debe ser comparada con las redes blockchain permisionadas.

Nombre	Tecnología	Pública/permisionada	Algoritmo de consenso	Transacciones/segundo
Bitcoin	Blockchain	Pública	Proof of Work	2-4
Ethereum	Blockchain	Pública	Proof of Work	5-7
Hyperledger Fabric	Blockchain	Permisionada	Byzantine Fault Tolerance	3.500
Hyperledger Sawtooth	Blockchain	Permisionada	Proof of Elapsed Time	100x
Alastria	Blockchain	Permisionada	Instambul BFT	100x
IOTA	Tangle	Pública	Markov Chain Monte Carlo	1.000.000
Hedera	Hashgraph	Permisionada	Gossip Protocol	250.000

Figura 2.18: Tabla resumen comparativa Blockchain.

Tras realizar un recorrido por algunos de las redes blockchain más representativas y de algunas tecnologías alternativas a blockchain resulta conveniente realizar una comparativa resumen de todas. La tabla XXXXXXX recoge las principales características de algunas de ellas.

Capítulo 3

Desarrollo de un software de análisis de trazabilidad de tiempo

3.1. Introducción

Como se describe en los objetivos del trabajo, una de las primeras tareas es conseguir un mecanismo con el cuál poder comparar dos relojes. Sin este mecanismo el trabajo carece de sentido ya que es vital conocer la precisión de la sincronización entre relojes. Denominado trazabilidad de tiempo, este mecanismo es a menudo usado por diversos grupos de trabajo relacionados con el tiempo y el posicionamiento para algunos propósitos específicos.

Pese a ser conocido para estos grupos, es necesario disponer de los dispositivos, software y bibliotecas requeridos para su correcto funcionamiento. Todo este procedimiento se recoge en el presente apartado.

3.2. Generación de ficheros RINEX

Entre otros, el método más completo para la generación de ficheros RINEX consiste en utilizar un dispositivo de posicionamiento GPS para la recepción de los datos. Este tipo de dispositivos están preparados para recibir las señales emitidas por las diversas constelaciones de satélites de los sistemas globales de navegación por satélite. La información recibida por dichos dispositivos se puede emplear para visualizar diversos parámetros como la localización o almacenar dicha información. Cada dispositivo grabará la información en un determinado formato propio y será necesario el uso de herramientas específicas para transformar dichos ficheros a un formato estandarizado si se necesita.

Figura 3.1: U-blox EVK-M8F [23].

3.2.1. Dispositivo U-blox

XXXXXXXXXXCambiar al ublox de JoseLuisXXXXXXXXXXXX El dispositivo U-blox EVK-M8F es un dispositivo de posicionamiento que cumple con el propósito anteriormente descrito.

El dispositivo se conecta al ordenador mediante USB del que recibe la alimentación. A su vez se conecta una antena que debe situarse en algún espacio al aire libre en el que la recepción de la señal sea apropiada. La conexión por USB al ordenador proporciona al mismo tiempo los datos que se reciben.

La empresa U-blox proporciona software para la correcta interacción con el dispositivo y los datos que este presta. U-center [24] es el software GNSS que soporta todos los dispositivos u-blox y permite visualizar e interactuar con los datos recibidos por este. Su interfaz gráfica permite visualizar los satélites de los que se recibe información, geolocalización, parámetros rela-

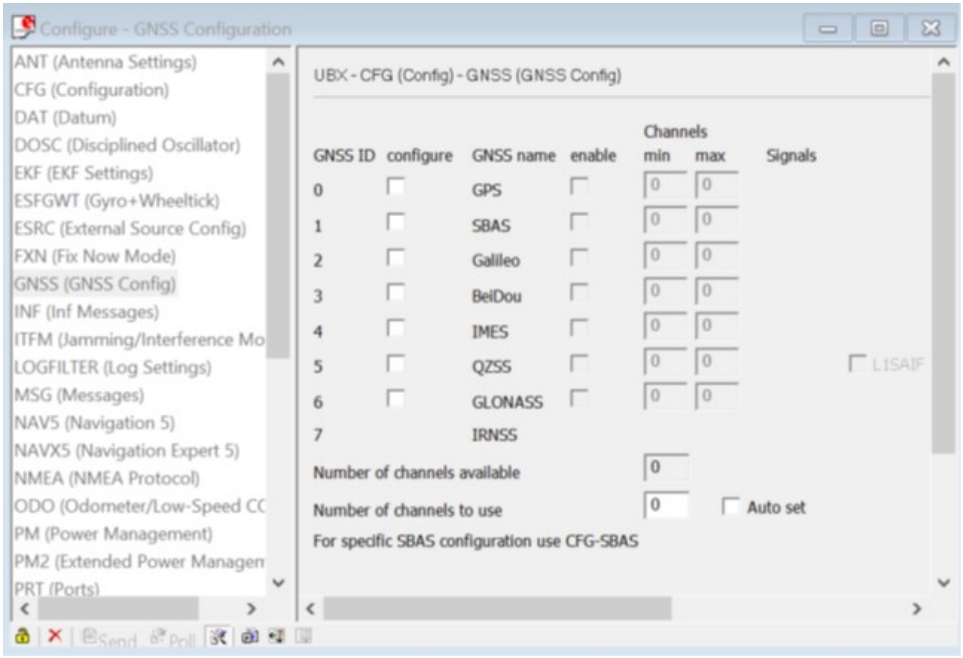

Figura 3.2: Configuración receptor U-blox desde u-center.

cionados con la señal recibida, el tiempo, etc. La ventana de configuración permite indicar al dispositivo los mensajes que se quiere que este transmita. También se puede configurar la constelación de la que se quiere que el receptor reciba y envíe información. Los posibles datos a recibir se engloban en dos tipos, (1) NMEA (National Marine Electronics Association), estándar para la interacción con los dispositivos marinos electrónicos y (2) UBX, protocolo binario para los mensajes de posicionamiento.

La información relevante para este trabajo se encuentra en los mensajes UBX-RXM-RAWX (Multi GNSS RAW Measurement Data). Estos mensajes lanzarán la información relacionada con el posicionamiento GNSS. Se activan por tanto estos mensajes y se procede al grabado de un fichero que contenga estos mensajes. Entre las herramientas se encuentra Record"que permite generar un fichero en formato ublox que almacena estos mensajes durante tanto tiempo como se desee. Notar que los mensajes se encuentran en formato raw en este fichero.

3.2.2. RTKLIB

La librería de aplicaciones para el posicionamiento GNSS contiene una serie de aplicaciones para interactuar con todo lo relacionado con este siste-

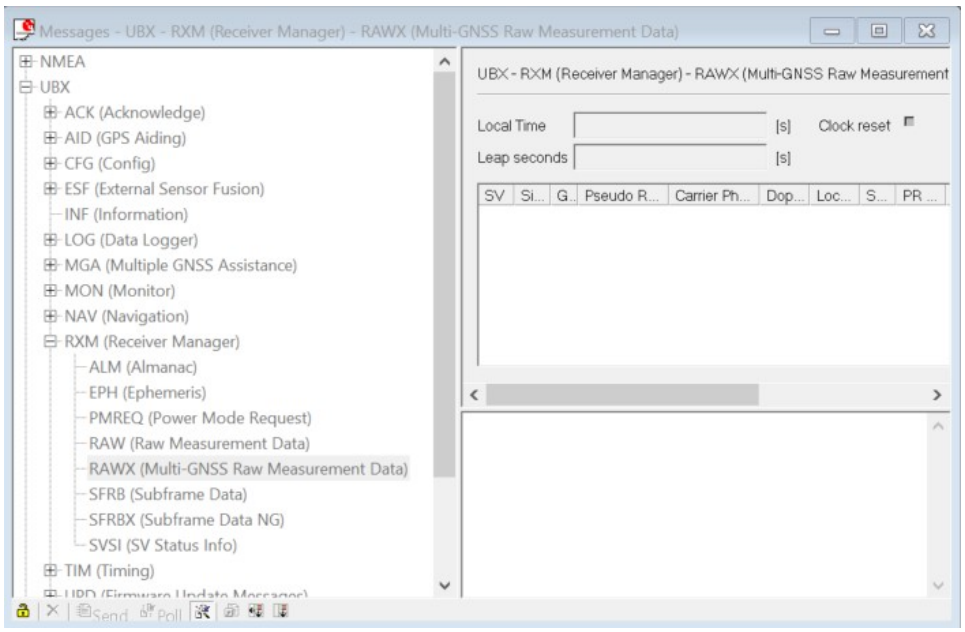

Figura 3.3: Ventana u-center para visualizar y habilitar mensajes.

ma.

En su repositorio de código [19] se encuentran los ejecutables de las distintas aplicaciones que ofrece. La aplicación CONVBIN permite convertir ficheros en diversos formatos a ficheros RINEX. Los ficheros generados en el apartado anterior sirven de entrada para esta aplicación. A la aplicación es necesario indicarle tanto el formato de entrada de los ficheros (ubx) como las salidas que se quiere que proporcione (ficheros de observaciones y de navegación).

La llamada a la aplicación se realiza por la terminal con el siguiente formato:

```
1 >> ./convbin.exe -r ubx ./rawData.ubx -o file.obs -n file.nav -g file.
    gnav
```

Una ejecución exitosa de la aplicación nos genera los ficheros necesarios para las próximas tareas.

3.2.3. Otros ficheros RINEX disponibles

Existen muchos grupos que proporcionan de manera abierta todos los ficheros RINEX que graban. Uno de estos grupos es el Royal Observatory of Belgium que almacena todos sus ficheros RINEX en un servidor FTP [9]

para que se puedan descargar de manera libre.

Se tiene en cuenta esta posibilidad ya que como se verá en el apartado de "Generación de ficheros CGGTTS", no todos los ficheros RINEX son válidos para generar un CGGTTS.

3.3. Generación de ficheros CGGTTS

El fichero CGGTTS contiene la información relacionada con la trazabilidad de tiempo del espacio temporal del que se tiene el fichero RINEX de un determinado dispositivo.

El software encargado de generar ficheros CGGTTS a partir de ficheros RINEX se llama R2CGGTTS. Este software es desarrollado, mantenido y distribuido por el grupo Royal Observatory of Belgium. Esta herramienta está disponible en el FTP del BIPM [8].

Al descargar la herramienta se observa que existen varias versiones de la misma. Durante los primeros meses de trabajo, la última versión disponible era la 7, así que al principio se trabaja con estas. En la documentación disponible para cada versión se observan las novedades que ha ido incluyendo cada una. Los aspectos más relevantes a tener en cuenta son la versión del fichero RINEX y las constelaciones GNSS que son soportadas.

Para comenzar a trabajar con la herramienta el primer paso consiste en compilar el código con un compilador Fortran 77.

```
1 >> gfortran R2CGGTTS_V7.f -o R2CGGTTS_executable
```

Una vez compilado y generado el ejecutable, ya está listo para su ejecución. Antes de ejecutarlo hay que tener en cuenta que (1) los ficheros RINEX se deben colocar en el mismo directorio que el ejecutable y (2) se deben rellenar dos ficheros especiales (paramCGGTTS.dat e inputFile.dat). Estos dos ficheros contienen los parámetros relacionados con el receptor y los nombres de los ficheros RINEX de entrada. Ahora ya se puede ejecutar el código. Desde una terminal habrá que desplazarse al directorio correspondiente y lanzar el ejecutable.

```
1 >> ./R2CGGTTS_executable
```

La salida esperada tras una ejecución exitosa es la creación en el mismo directorio de los ficheros CGGTTS.

3.3.1. Problemas con la herramienta

Pese a que fueron varios los problemas acontecidos durante el trabajo, hay dos que son los más representativos y que merece la pena mencionar.

3.3.1.1. Versión del fichero RINEX

Cada versión nueva del software R2CGGTTS ha ido incluyendo novedades a lo largo del tiempo. Una de ellas fue el cambio de versión (de la versión 2.x a la 3.x) de los ficheros RINEX entre las versiones 5 y 7 de la herramienta. El procedimiento seguido para generar un fichero RINEX (descrito en el apartado "Generación de ficheros RINEX") genera un fichero RINEX de versión 2.11. Esta versión es soportada solamente por la versión 5 del software R2CGGTTS. En paralelo a esto se prueba la versión 7 de R2CGGTTS con ficheros RINEX versión 3 disponibles online para su descarga.

En este tiempo se hace uso de herramientas que permiten la conversión de versiones de los ficheros RINEX [11]. Algunas de estas herramientas también permiten la conversión de ficheros raw en formato ublox a la versión deseada de fichero RINEX. Sin embargo, de ninguna herramienta se obtiene la salida esperada.

Durante este tiempo de trabajo aparece la versión 8 del software R2CGGTTS. Entre las novedades que incluye, la principal consiste en permitir todas las versiones de ficheros RINEX. Cuando se subsana este problema aparece el siguiente que merece la pena mencionar.

3.3.1.2. Requisito GPS P1 code o fichero biasC1P1.dat

En la documentación, en el apartado de ficheros de entrada aparece un requisito: "*biasC1P1.dat :Nedded if GPS P1 code is missing in Rinex observation file*". En caso de que el fichero RINEX no disponga de esa información, se necesitará un fichero adicional.

Tras investigar sobre el tema, se llega a la conclusión de que la información *GPS P1 code* es dependiente del receptor. Con el receptor U-blox XXXXXXXXXXXXXJoseLUis no es posible crear un RINEX que contenga esta información, sólo ciertos receptores lo permiten.

XXXXXXXXXXLista de receptores P1XXXXXXX

En el apartado "Generación de ficheros RINEX"se menciona un servidor FTP en el que el ROB almacena diariamente sus ficheros RINEX de manera que estén disponibles para su descarga. Los receptores utilizados por este grupo son bastante más avanzados y permiten crear ficheros RINEX con toda la información necesaria. Como plan alternativo, se hace uso de estos fichero RINEX para comprobar que efectivamente con esos ficheros el software R2CGGTTS se ejecuta satisfactoriamente y genera los ficheros CGGTTS esperados.

Capítulo 4

Desarrollo de software confiable

4.1. Introducción

En este apartado se describe el procedimiento seguido hasta conseguir implementar TEE. Al principio se hace uso de la tarjeta Zedboard de Xilinx. Sin embargo, surgen varios inconvenientes a medida que avanza el trabajo, que sumados a algunos requisitos del trabajo provocan que se deba cambiar de dispositivo. El dispositivo que se acaba usando es una Raspberry Pi 3.

El procedimiento seguido para desarrollar una aplicación de ejemplo sobre TrustZone en Zedboard resulta muy representativa de lo que se consigue con esta tecnología de ARM. Se incluye dicho procedimiento de manera detallada ya que en Raspberry Pi se hace uso del proyecto OP-TEE que realiza el procedimiento de manera transparente.

4.2. ARM TrustZone en Zedboard

El primer dispositivo sobre el que se pretende desarrollar un diseño con TrustZone habilitado es la tarjeta Zedboard de Xilinx. La información práctica existente proporcionada por ARM o Xilinx de manera abierta sobre TrustZone es muy reducida. La documentación que proporcionan trata de manera teórica el tema, sin embargo, las guías y tutoriales prácticos (principalmente los de ARM) son de pago o bajo acuerdos. Debido a esto, se propone en un primer momento realizar un diseño de ejemplo que sirva de aprendizaje y comprobación del correcto funcionamiento de TrustZone.

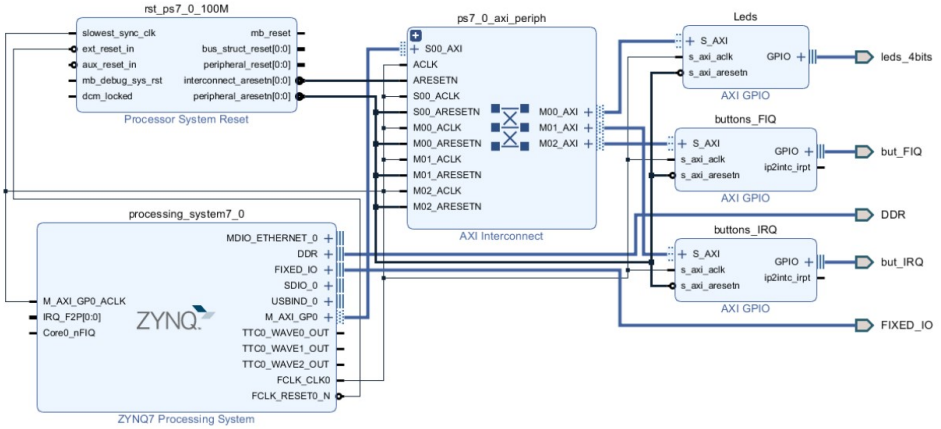

Figura 4.1: Diseño de bloques final.

4.2.1. Diseño TrustZone Hello World

Para este diseño se hace uso del software *Vivado Design Suite* de Xilinx. La principal guía de usuario proporcionada por Xilinx se titula *Programming ARM TrustZone Architecture on the Xilinx Zynq-7000 All Programmable SoC* [16]. En esta guía aparecen reflejados los cambios y configuración necesarios en registros, señales, memoria, etc. A su vez se hace uso de un tutorial [26] que se basa en la guía de Xilinx. Esta documentación es la base para desarrollar el diseño de ejemplo.

El diseño consiste en configurar como seguro un GPIO y desarrollar dos aplicaciones *Hello World* (una en el Mundo Seguro y otra en el Mundo Normal) que se llamen una serie de veces la una a la otra mediante el Monitor.

Los primeros pasos a seguir son los convencionales, se crea un proyecto para la tarjeta Zedboard y se crea el diseño de bloques deseado. En este caso consiste del Sistema de Procesamiento ZYNQ, 3 GPIOs y los correspondientes bloques (como el AXI de Interconexión) que se crean cuando se realiza la conexión automática de ayuda de Vivado. En la figura XXXXX se observa el diseño final.

4.2.1.1. Configuración del diseño

En el diseño se configuran dos aspectos relevantes para TrustZone:

Interrupciones IRQ y FIQ. Como aparece descrito en la guía de usuario de Xilinx [16] en el apartado *Securing Interrupts*, el controlador de in-

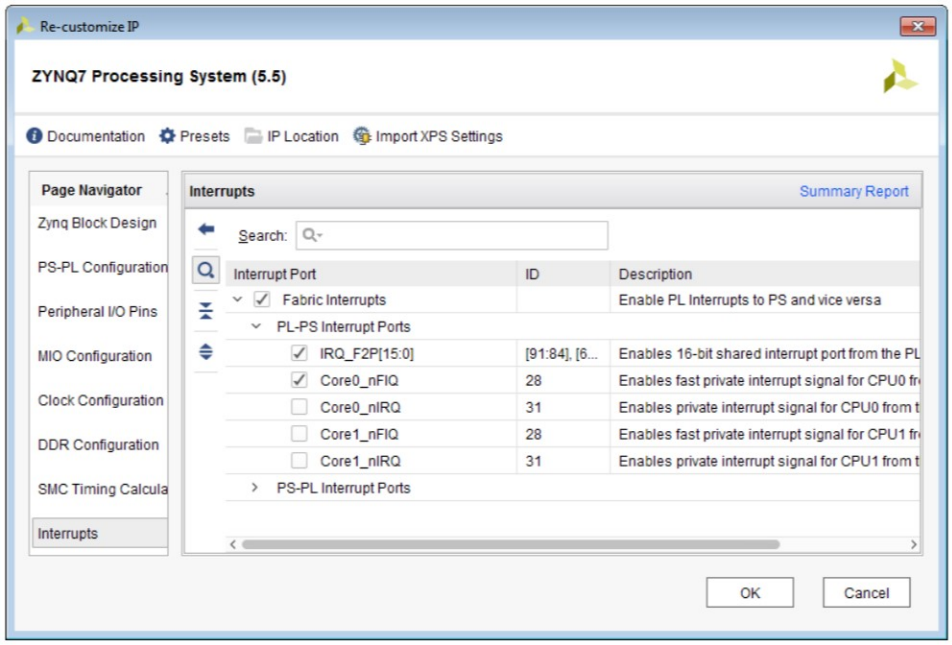

Figura 4.2: Habilitar interrupciones IRQ y FIQ.

terrupciones del ARM Cortex A9-MP permite definir individualmente las interrupciones como seguras o no seguras. Las interrupciones no seguras se suelen señalizar haciendo uso del mecanismo de interrupciones IRQ. En cambio, las interrupciones seguras pueden utilizar tanto el mecanismo IRQ como el FIQ. En la configuración del procesador, en el apartado interrupciones, aparece la opción de habilitar tanto IRQ como FIQ. Una vez se habilitan como se observa en la figuraXXXXXXXXXX, ambas deben aparecer en el bloque del procesador para poder ser conectadas (se observan en la figura XXXXXXanterior).

Securizar periféricos. En el diseño, M0X_AXI es el máster que controla los periféricos (GPIOs). Para conseguir que uno de esos periféricos sea seguro hay que habilitar la opción *Secure Slave* del bus encargado de la comunicación entre el *AXI Interconnect* y el periférico en cuestión. En la configuración del *AXI Interconnect* en Opciones Avanzadas existe la posibilidad de configurar alguna de estas interfaces (M0X_AXI) como seguras. En la figura XXXX se observa el procedimiento seguido para hacer segura la interfaz M00_AXI encargada de controlar el GPIO denominado Leds.

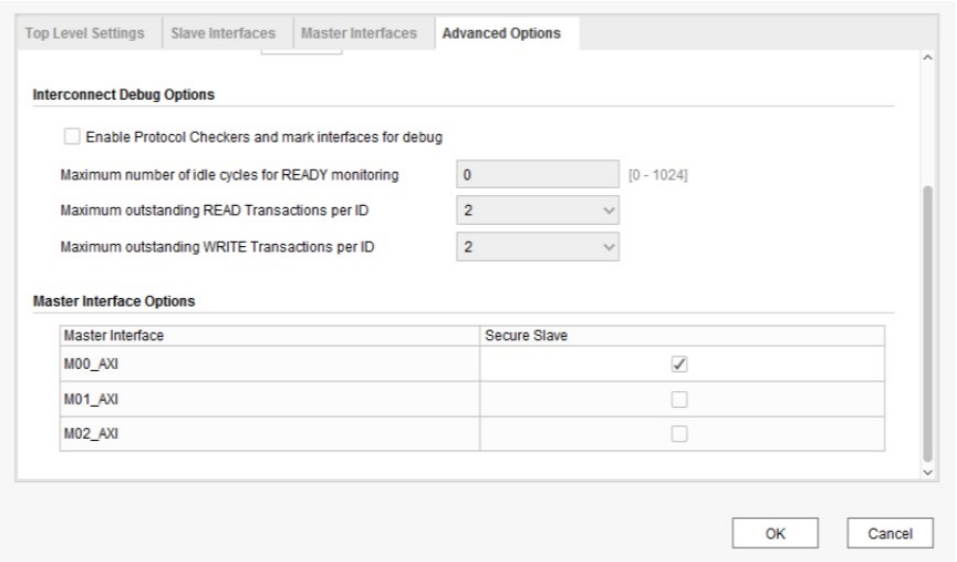

Figura 4.3: Securización de la interfaz del periférico.

4.2.1.2. Implementación software

Para esta tarea se hace uso del *Software Development Kit* incluido en Vivado. Se exporta el proyecto al SDK para poder empezar a trabajar en él.

Antes de poder comenzar a desarrollar el código de la aplicación de ejemplo, hay una serie de archivos del proyecto que hay que modificar e incluir algún otro para el correcto funcionamiento de TrustZone. Notar que estos archivos se encuentran tanto en el código ejemplo proporcionado por ARM *Cortex-A9 TrustZone example* [7], como en el tutorial [26]:

Creación de proyectos. Se pretender desarrollar una aplicación segura y otra no segura. Por lo tanto, será necesario crear dos proyectos que contentan una aplicación con la plantilla *Hello World*.

Archivo *lscript.ld*. Estos archivos contienen las direcciones de las distintas regiones de memoria para cada aplicación. Se modifican tanto la dirección de origen como el tamaño de manera conveniente a cada aplicación. En el caso de la aplicación segura, hay que añadir una sección para la imagen de la aplicación no segura y la pila del monitor (consiste en dos particiones una para el Mundo Seguro y otra para el Mundo Normal).

Archivo *monitor.S*. Este archivo contiene la rutina para el *Secure Monitor Call*. Su principal función es cambiar de un Mundo (Seguro/Normal) a

otro, almacenando los registros de memoria del mundo en el que se encuentra y cargando los del mundo al que cambia.

Archivo *normal.S.* Este archivo es el encargado de enlazar con la imagen del Mundo Normal desde el Mundo Seguro. Esto es posible ya que se incluye el binario de la aplicación no segura en la aplicación segura.

Archivo *platform.c.* En este archivo se deben de incluir algunas líneas encargadas de (1) la definición de algunos macros para los registros TrustZone y (2) incluir la llamada a función de inicialización de TrustZone (*init_TZ()*) a la función *init_platform()*.

Archivo *boot.S.* Este archivo propio de Xilinx se encuentra en la ruta *non_secure_bsp/ps7_cortexa9_0/libsrc/standalone_v6_2/src/*. Dicho archivo es un *standalone* de Xilinx al que hay que realizar algunos cambios en la configuración del registro CP15 para que no se produzcan excepciones indeseadas.

Una vez hecho esto se puede proceder a desarrollar el código de las aplicaciones (segura y no segura). Se parte de la plantilla existente para una aplicación de ejemplo Hello World. Se realizan una serie de cambios para que el comportamiento sea que una aplicación llame a la ejecución de la otra mediante llamadas al SMC. El código resultante se observa en las figuras XXXXXX y XXXXXXXXXXXXX.

Efectivamente se obtiene el comportamiento esperado, 10 mensajes "Hello World"(5 del Mundo Seguro y 5 del Mundo Normal) se imprimen en una consola conectada por serial (115200 Baudrate) a la UART de Zedboard.

4.2.2. Problemas para el diseño deseado

El diseño deseado para el presente trabajo tiene una serie de requisitos que, como se verá a continuación, ocasionarían varios inconvenientes en Zedboard. Principalmente la autenticación del código es el principal motivo que ocasiona el cambio de dispositivo.

4.2.2.1. Autenticación del código

El diseño planteado en el apartado anterior resulta muy representativo para el funcionamiento de TrustZone. Sin embargo, para el diseño de un sistema seguro es necesario dar un paso más, principalmente en el arrancado del sistema. TrustZone asegura que la ejecución de la apliación segura desarrollada y la comunicación con el periférico sean seguras. Existen ataques que se llevan a cabo cuando el dispositivo está apagado, alterando el código

```c
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"

#define SMC __asm__("SMC #0")

extern void monitorInit();

int main()
{
    init_platform();

    /* Monitor mode initialization */
    monitorInit();

    for (unsigned char i = 0; i < 5; i++)
    {
        /* Print a message then move to the normal world */
        xil_printf("Hello from secure world !\n");
        SMC;
    }

    cleanup_platform();
    return 0;
}
```

Figura 4.4: Código de la aplicación segura.

```c
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"

#define SMC __asm__("SMC #0")

int main()
{
    /* Move to the secure world */
    SMC;
    for (unsigned char i = 0; i < 5; i++)
    {
        /* Print a message then move to the secure world */
        xil_printf("Hello from non_secure world !\n");
        SMC;
    }

    return 0;
}
```

Figura 4.5: Código de la aplicación no segura.

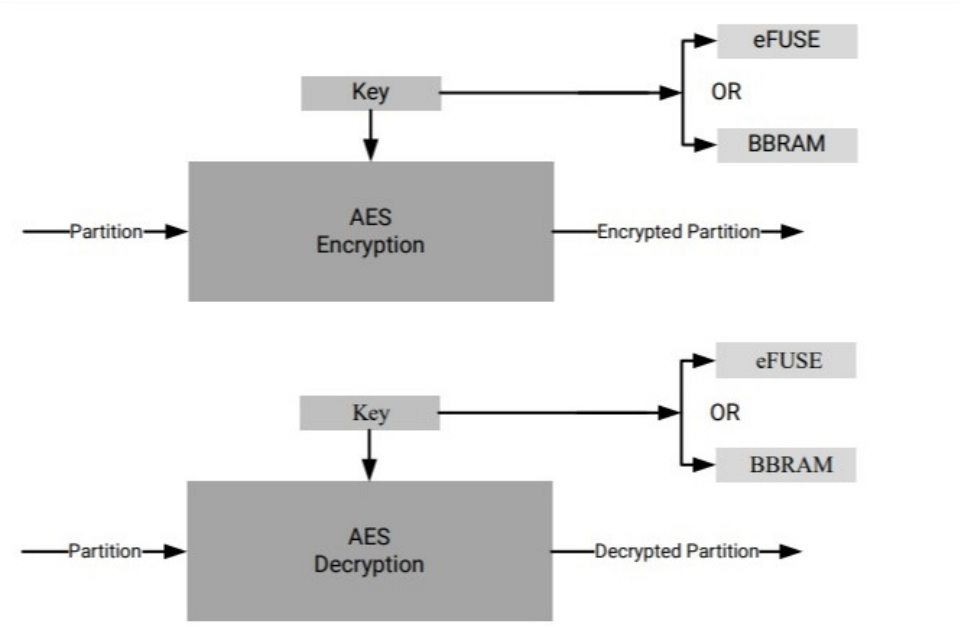

Figura 4.6: Cifrado y descifrado de una partición.

o la imagen que posteriormente se pondrá en marcha. Además, se pretende que el sistema desarrollado se lleve a cabo en una red distribuida, por lo tanto, es necesaria la autenticación del código o imagen previa al arranque del sistema.

La autenticación del código consiste en el uso de criptografía asimétrica para el cifrado del código. El código a autenticar se cifra con la clave privada para que posteriormente el dispositivo lo descifre. Para que el dispositivo pueda descifrar el código es necesario que contenga la clave pública (en realidad solo contiene un hash de la clave pública) como se observa en la figura XXXXX que muestra el proceso de cifrado y descifrado. Se necesita que esa clave pública no pueda ser inalterada, ya que en caso contrario se podría autenticar cualquier otro código que no sea el deseado. Para este propósito se utilizan los registros OTP (One Time Programmable) del dispositivo. En Zedboard los registros OTP son los eFUSE o los BBRAM (Battery Backed RAM). La tarjeta Zedboard no lleva batería incrustada, por lo que para hacer uso de estos registros es necesario conectar una fuente de tensión externa [6]. En la fase de fabricación del dispositivo se graba la clave pública en los registros OTP de manera que esta no pueda ser alterada.

Este procedimiento es el seguido para un entorno de producción. En el caso de un entorno de desarrollo se utiliza una técnica distinta que consiste

en insertar la clave pública en la cabecera del código de arranque (Boot Header) en lugar de en los registros OTP. Para utilizar esta técnica, es necesario incluir una serie de atributos al fichero BIF (Boot Information File) utilizado para crear el binario de arranque (*boot.bin*) [4]. Desafortunadamente dichos atributos solamente son soportados para dispositivos Zynq UltraScale+ MPSoC, entre los cuales no se encuentra la Zedboard.

Al mismo tiempo que se encuentra este problema se descubre el proyecto OP-TEE. Dicho proyecto realiza tanto la parte relacionada con TrustZone como la autenticación de manera automática. La integración de dicho proyecto supone un gran paso con el único inconveniente del cambio de Zedboard a Raspberry Pi 3.

4.3. ARM TrustZone en Raspberry Pi 3

El principal motivo del cambio de Zedboard a Raspberry Pi 3 se debe al proyecto OP-TEE. El objetivo del proyecto OP-TEE además de tener un objetivo que encaja a la perfección con los requisitos del trabajo, es un proyecto muy activo que intenta solucionar todos los *Issues* que se publican en su repositorio.

El propósito de OP-TEE es principalmente educacional. Esto se debe a que no proporciona un sistema seguro útil en un entorno de producción, solamente para entornos de desarrollo. Raspberry Pi 3 proporciona un procesador que soporta ARM TrustZone, sin embargo, no proporciona registros OTP. La diferencia con Zedboard es que en este caso se puede hacer uso de la técnica que utiliza el Boot Header para el arranque seguro. La clave pública se inserta en este Boot Header para posteriormente poder realizar el descifrado .[a]utenticando.[el] código.

4.3.1. Construcción del entorno seguro con OP-TEE

En el repositorio oficial del proyecto OP-TEE [14] se proporcionan los archivos necesarios para compilar todos los componentes necesarios (kernel de Linux, ARM Trusted Firmware, etc). Sin embargo, para facilitar la tarea proporcionan una serie de *Makefiles* y *shell-scripts* que realizan todo ese trabajo de manera autónoma.

Los comandos a seguir para dicho proceso se recogen a continuación:

1. Instalar una serie de paquetes requeridos previamente.

```
$ sudo apt-get install android-tools-adb android-tools-fastboot
    autoconf automake bc bison build-essential cscope curl device
    -tree-compiler  expect flex ftp-upload gdisk iasl libattr1-
    dev libc6:i386 libcap-dev libfdt-dev libftdi-dev libglib2.0-
    dev libhidapi-dev libncurses5-dev libpixman-1-dev libssl-dev
    libstdc++6:i386 libtool libz1:i386 make mtools netcat python-
    crypto python-serial python-wand unzip uuid-dev xdg-utils
    xterm xz-utils zlib1g-dev git
```

2. Instalar el repositorio Android y configurar git.

```
$ mkdir ~/bin
$ PATH=~/bin:$PATH
$ curl https://storage.googleapis.com/git-repo-downloads/repo >
    ~/bin/repo
$ chmod a+x ~/bin/repo
$ git config --global user.name "Your Name"
$ git config --global user.email "you@example.com"
```

3. Descargar el código de OP-TEE y compilar.

```
$ mkdir -p $HOME/devel/optee
$ cd $HOME/devel/optee
$ repo init -u https://github.com/OP-TEE/manifest.git -m rpi3.xml
$ repo sync
$ cd build
$ make toolchains
$ make
$ make img-help
```

   Este último comando muestra por pantalla los pasos a seguir para realizar las particiones de la tarjeta SD y que cada una contenga los archivos correspondientes.

4.3.1.1. Linux customizado con Buildroot

El kernel Linux que crea el proyecto por defecto contiene una serie muy limitada de paquetes. El objetivo de esto es que la imagen sea tan pequeña como sea posible. Sin embargo, en el caso de que se necesite un sistema operativo con ciertos paquetes se pueden realizar los cambios pertinentes para que la compilación los incluya directamente.

La herramienta que usa OP-TEE para generar Linux es Buildroot [5]. En su repositorio se encuentran todos los paquetes que se pueden incluir. El procedimiento para añadirlos en el proyecto OP-TEE consiste en modificar el archivo *common.mk* incluyendo una linea con el nombre del paquete deseado. En este caso se incluye el paquete PTPd con la siguiente línea:

```
...
buildroot: optee-os
@echo "BR2_PACKAGE_PTPD=y" >> ../out-br/extra.conf
...
```

Tras este cambio será necesario volver a ejecutar el comando *make* en el directorio *build* y volver a volcar los archivos sobre la tarjeta SD. El paquete PTPd está ahora disponible para su uso.

Capítulo 5

Propuesta Blockchain basada en tiempo confiable

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

En base al protocolo actual que se tiene para las redes blockchain basadas en prueba de trabajo, ninguna de estas puede ser un sistema escalable. Si se quiere que una red blockchain basada en prueba de trabajo siga teniendo las características necesarias para sustentar una moneda digital, no se podrá conseguir que el sistema sea escalable como tal. Está diseñado para que sea ineficiente. Sin embargo, es posible hacer cambios en el protocolo para que, dentro de esa ineficiencia, el sistema sea lo más eficiente posible.

¿Por qué está diseñado para ser ineficiente? La prueba de trabajo plantea problemas matemáticos a toda la red de usuarios de manera que el primero en encontrar la solución, la distribuya, sea el creador del siguiente bloque de la cadena y se lleve una recompensa por ello. El problema está en la distribución de la solución. Se necesita que la mayoría de la red esté de acuerdo con que un usuario en concreto ha sido el primero en encontrar la solución a la prueba de trabajo y todo lo que ello acarrea. El consenso debe llegar antes de que los usuarios de la red comiencen a buscar solución a la nueva prueba de trabajo. Los usuarios forman una red P2P, por lo que la distribución de la solución llevará su tiempo antes de que la mayoría de la red alcance el consenso. Para este tiempo es para el que se ajusta la dificultad de la prueba de trabajo, y no se puede reducir ya que sino la red no alcanzaría el consenso. En la red Bitcoin este tiempo es de aproximadamente 10 minutos. Suele ocurrir que, durante este tiempo de difusión de una solución, haya otros usuarios que, sin saber que alguien ya ha encontrado una solución, encuentren otra solución al mismo problema y comiencen a distribuirla desde otro punto de la red. Habrá usuarios de la red que reciban soluciones a la misma prueba de trabajo de distintos usuarios. Sólo una de ellas se puede

considerar válida, y los usuarios deben ponerse de acuerdo en cuál de ellas. Cada vez que se encuentra una solución a la prueba de trabajo, a su vez se genera una nueva prueba de trabajo relacionada con esa solución anterior. El protocolo actual dicta que cuando un usuario recibe varias soluciones válidas a una misma prueba de trabajo, este debe elegir una de estas soluciones al azar y comenzar a buscar solución a la nueva prueba de trabajo generada. Cada usuario espera que la mayoría de la red haya elegido la misma solución que él, ya que sino estará buscando solución a una nueva prueba de trabajo distinta a la de la mayoría de la red y por ende inválida. Cuantas más soluciones se encuentren a la misma prueba de trabajo, más difícil resultará alcanzar el consenso. Por lo tanto, la dificultad de la prueba de trabajo se ajustará para que el número de soluciones concurrentes sea el mínimo posible. A priori se puede pensar que resultaría mucho más eficiente que cada usuario sellara temporalmente su solución. Así cuando otro usuario de la red tenga que elegir entre varias soluciones, podrá hacerlo en base al sello temporal y no por azar. Esto da pie a otro de los puntos clave de las redes blockchain, y es que, al tratarse de redes P2P públicas, ningún usuario debe confiar en ningún otro. Por lo tanto, hay dos aspectos de los que ningún usuario debe fiarse:

1. No se tiene la certeza de que otro usuario selle temporalmente una solución con la hora real de su reloj. Podría hacerlo con una hora previa para sacar ventaja de ello.

2. No se tiene la certeza de que la hora real que marque el reloj de un usuario esté sincronizada con la hora UTC.

El objetivo del presente trabajo consiste en solventar estos dos puntos. De conseguirlo, se podría hacer uso de un sellado temporal distribuido y confiable para las distintas soluciones que se encuentren a la prueba de trabajo. Se considera que esto permitiría a los usuarios de la red encontrar el consenso de manera más eficiente. Así se podría reducir la dificultad de la prueba de trabajo, sin que esto afecte al consenso, lo que supondría una mejora en el número de transacciones por segundo.

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

En el apartado de ObjetivosXXXXXXXXXXXXXXX se describen los principales problemas que tienen las redes blockchain públicas basadas en prueba de trabajo. El presente trabajo tiene como objetivo atajar el problema de la escalabilidad, no para hacer de estas redes blockchain un sistema escalable como tal, sino para intentar sacar el mayor rendimiento posible. El principal factor limitante para la escalabilidad nace de la prueba de trabajo. Sin embargo, ya se ha visto por qué la prueba de trabajo no se puede facilitar si se quiere que la red mantenga el consenso. La propuesta de este trabajo consiste en un sistema que permita hacer la prueba de trabajo más

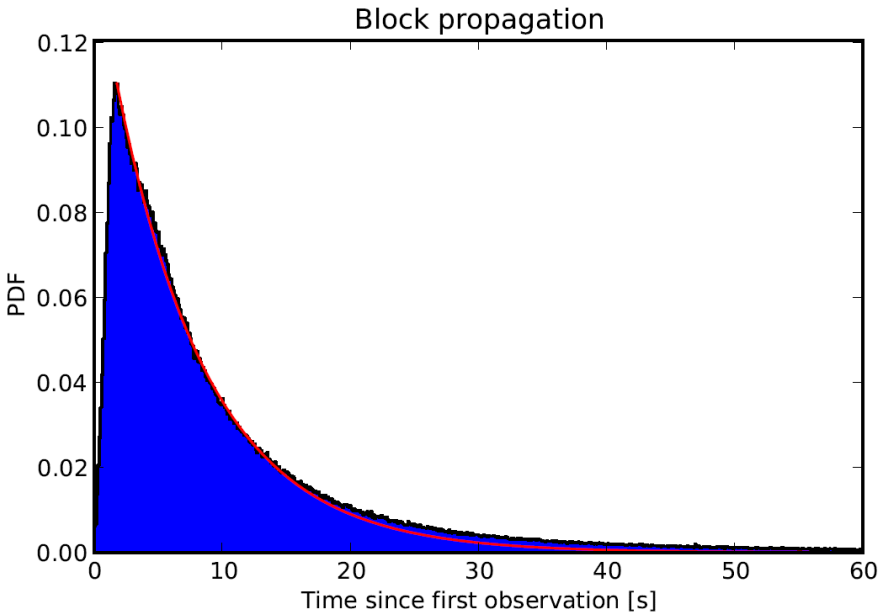

Figura 5.1: Tiempo de propagación de un bloque [29].

fácil manteniendo el consenso de la red. Para esto es necesario ahondar más en las consecuencias para comprender la propuesta.

5.1. Forks en blockchain

Para entender completamente en qué consiste un fork y las consecuencias que estos pueden acarrear en las cadenas de bloques se hace uso de un paper que estudia la propagación de los bloques por la red Bitcoin [29].

En el estudio se analiza en un primer momento el retardo de la red blockchain de Bitcoin. Se denomina retardo D al tiempo que tarda un bloque emitido por un minero en llegar al resto de la red. En la figura XXXXXXXXXXX se observa el resultado obtenido tras analizar 10000 bloques de la cadena. El retardo medio es de 12.6 segundos. Pasados 40 segundos aún hay un 5 % de los nodos que no han recibido el bloque. El objetivo de medir este retardo es ver su interacción real con el número de forks que se suceden en la red. Cuando el primer minero de la red consigue minar el bloque de altura (o longitud) N+1, inmediatamente lo emite a la red. Desde esta emisión transcurre un tiempo hasta que el resto de mineros de la red recibe este bloque. Durante este tiempo denominado D, el resto de los nodos de la red siguen intentando minar el bloque N+1 ya que aún no se han enterado de que otro minero ya ha conseguido minarlo. Si dentro de este tiempo D algún

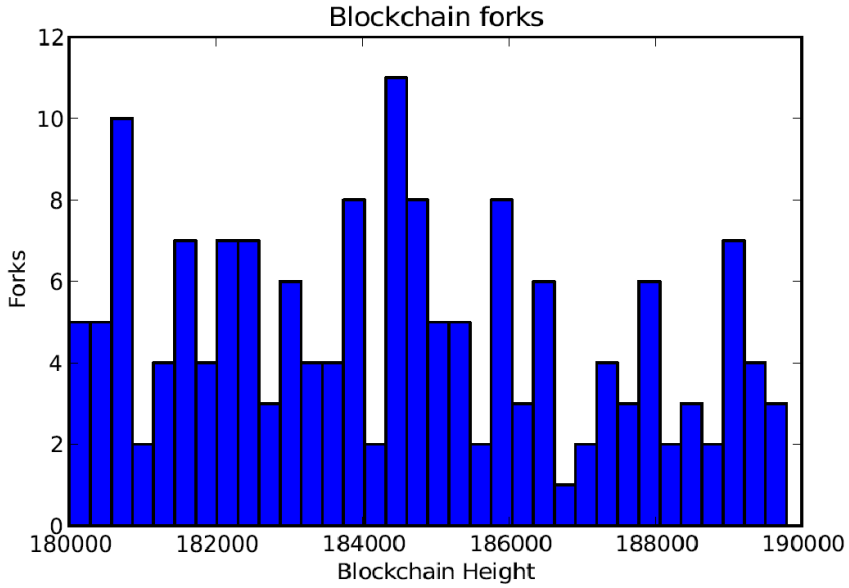

Figura 5.2: Nº de forks con nodo en modo pasivo [29].

otro minero consigue minar otro bloque N+1, se tendrán 2 bloques N+1 emitidos a la red. Estos dos bloques se denominan (N+1) y (N+1)'. En este momento se podría dar el caso de que una parte de la red tome como válida la cadena que acaba con el bloque (N+1) y otra parte tome la cadena que acaba con el bloque (N+1)'. Habría que esperar al minado de varios bloques más para que la red se ponga de acuerdo en una única cadena, suponiendo esto un desperdicio de poder computacional y tener que revertir una cadena de longitud considerable (PUEDE QUE NO SE ENTIENDA LO DE REVERTIR CADENA QUITARLO???XXXXXXXXXXXXX). Cuanto mayor sea el tiempo D mayor es la probabilidad de que se produzcan forks en la red.

El estudio pasa a medir el número de forks que se observan en un intervalo de 10000 bloques. Para ello monta un nodo "pasivo" que está conectado a una gran cantidad de nodos (bastante más grande de lo normal) con el objetivo de observar la mayor parte de la red posible. En la figura XXXXXXXXXX se observa que se producen un total de 169 forks en los 10000 bloques (r=1.69 %). En un intento por demostrar la repercusión del tiempo D en los forks producidos en la red, realizan un experimento en el que aumentan la conectividad de la red. Para ello hacen uso del anterior nodo con gran cantidad de nodos conectados, pero en modo "activo" (en este caso actuará en base al protocolo haciendo broadcast de la cadena más larga que conozca en cada momento). De nuevo miden durante 10000 bloques el número de forks, los resultados se observan en la figura XXXXXXXXXXXXx.

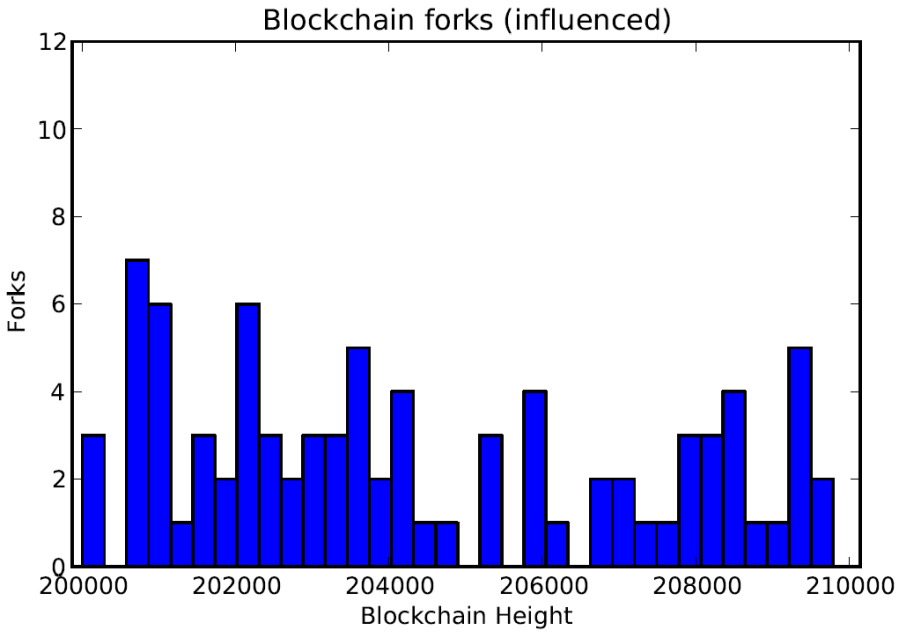

Figura 5.3: Nº de forks con nodo en modo activo [29].

La tasa de forks pasa de un 1.69 % a un 0.78 %, esto supone una mejora de un 53.41 % en el número de forks. La conclusión que se obtiene de este estudio es que aumentar la conectividad de un nodo (400 conexiones frente a las 8 que suele tener un nodo) ha provocado un descenso considerable en la tasa de forks. Esto se ha producido por el simple hecho de que se ha conseguido alcanzar el consenso en una misma cadena más rápido que si se tuviese que esperar a que alguna cadena adelante a otra (en número de bloques). Si un nodo ha recibido 2 cadenas bien formadas distintas de la misma longitud, tendrá que elegir una al azar. Si muchos nodos eligen con qué cadena quedarse al azar el resultado será que no haya consenso hasta más adelante. Este es el principal foco de atención de este trabajo, conseguir que ningún nodo tenga que elegir al azar con qué cadena quedarse en ningún momento.

El número de forks no solamente es un problema para el consenso, sino también para la seguridad. Para ver por qué, se hace uso de un artículo hecho para la red Bitcoin [34] que explica por qué una disminución del tiempo de la prueba de trabajo (que aumente la tasa de transacciones por segundo), que se traduciría en un incremento de los forks, haría la blockchain más vulnerable.

La figura XXXXXXXXXXx muestra una situación en la que un atacante intenta realizar un ataque del 51 %. Se supone que el atacante tiene un

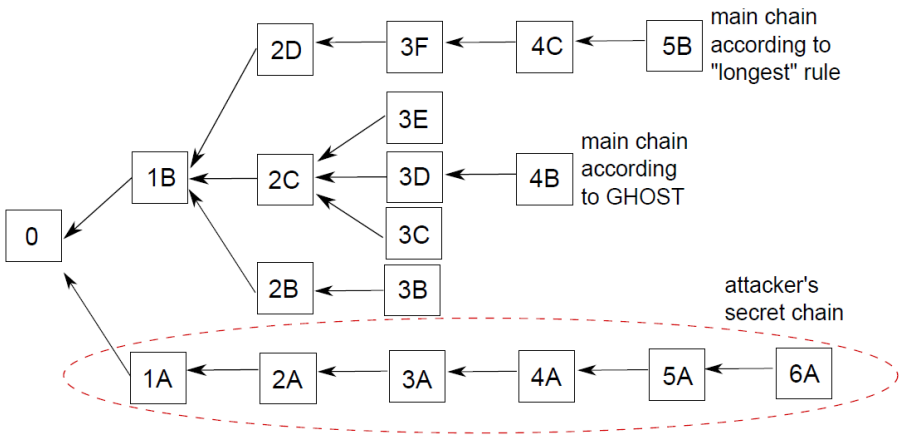

Figura 5.4: Estrategia "Self-Mine-Strategy" [34].

poder computacional del 30 % y el resto de la red honesta dispone del 70 % restante. Disminuir el tiempo de la prueba de trabajo conlleva un mayor número de forks. Cuando se producen varios forks, la parte honesta de la red está divida en tantas partes como forks haya. A la altura del bloque número 2 de la figura XXXXXXXxxxx se tienen 3 forks, por tanto, el 70 % del poder computacional de la red honesta estará dividido en estos 3 forks. Cada parte de la red estará intentando minar el siguiente bloque de la cadena sobre una de las 3 cadenas distintas que se tiene. El atacante está actuando por su cuenta, así que emplea el 30 % de su poder computacional íntegramente en minar sobre su única cadena. A esta técnica de minado del atacante se la denomina "Self-Mine-Strategy". Es evidente que el atacante con sólo el 30 % del poder computacional de la red será capaz de crear una cadena más larga que el resto de la red honesta y llevará su ataque con éxito.

Esta situación se podría haber evitado con la propuesta que presenta este trabajo. Si se observa con detenimiento la figura XXXXXXXXXXXX, se ve que los forks que se han producido a la altura de los bloques 2 y 3 se debe a que los nodos han recibido varias cadenas de igual longitud y han tenido que elegir una de ellas al azar. Si se consigue evitar esta situación en la que los nodos no sepan qué cadena de igual longitud (en número de bloques) elegir, se mantendría el consenso en una misma cadena. Manteniendo el consenso en una sola cadena haría que toda la parte honesta de la red volcase todo su poder computacional (70 %) en la misma cadena y el atacante con su 30 % no pueda llevar a cabo el ataque.

La situación idílica en el caso de que se produzca un fork sería no sólo que todos los nodos que están en esta situación elijan la misma cadena, sino que todos elijan la primera que realmente se minó. A priori se podría pensar que, si el minero añadiese un timestamp al bloque al minarlo, el resto podría saber fácilmente qué cadena se minó la primera. El timestamp en blockchain es un tema algo controvertido que merece la pena tratar, sobre todo para la propuesta del trabajo.

5.2. El papel del timestamp en blockchain

Una situación que a menudo es confusa de blockchain es la puntual falta de orden temporal del timestamp de los bloques. En repetidas ocasiones un bloque posee un timestamp menor al timestamp de un bloque previo. En base a la mayoría de los protocolos blockchain actuales, esta situación es totalmente válida.

Una red blockchain no posee una autoridad central que informe del tiempo UTC (Universal Time Coordinated). Asumir que el tiempo local de cada nodo es preciso o está sincronizado con los relojes del resto es un error. Cada nodo está conectado a la red a través de una serie de pares. Cada nodo posee un tiempo UTC local con un offset marcado por la media del tiempo UTC local de cada par (peer?????XXXX) a los que está conectado. El tiempo global de la red blockchain está marcado por la media de los timestamps emitidos por todos y cada uno de los nodos de la red. Esto provoca que la sincronización en estas redes no sea nada óptima, provocando situaciones en las que el orden de los timestamps no sea correcto.

El protocolo de Bitcoin únicamente rechaza un timestamp si es menor a la media de los timestamps de los 11 bloques anteriores o 2 horas superior al tiempo global de la red. De manera parecida se realiza en Ethereum, provocando que un nodo pueda falsear su timestamp en unos 900 segundos sin repercusión alguna.

Bibliografía

[1] Amd guardmi technology. https://www.amd.com/en/technologies/guardmi.

[2] Bitcoin repository. https://github.com/bitcoin/bitcoin/tree/master/src.

[3] Blockchain explorer. https://www.blockchain.com/explorer.

[4] Bootgen user guide. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_2/ug1283-bootgen-user-guide.pdf.

[5] Buildroot. making embedded linux easy. https://buildroot.org/.

[6] Changing the cryptographic key in zynq-7000 ap soc. https://www.xilinx.com/support/documentation/application_notes/xapp1223-crypto-key-change.pdf.

[7] Cortex-a9 trustzone example. http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.faqs/ka15417.html.

[8] Ftp of bureau international des poids et measures. ftp://tai.bipm.org/soft/r2cggtts.

[9] Ftp of royal observatory of belgium. ftp://gnss.oma.be/gnss/data/rinex/daily/.

[10] Gartner hype cycle: Bitcoin. https://www.gartner.com/technology/research/methodologies/hype-cycle.jsp.

[11] Gnss converter. https://process.gps-solutions.com/index.html.

[12] The hedera hashgraph platform. https://www.hedera.com/whitepaper.

[13] Intel software guard extensions. https://software.intel.com/es-es/sgx.

[14] Open portable trusted execution enviroment. https://github.com/OP-TEE.

[15] Precision time protocol daemon. https://github.com/ptpd/ptpd.

[16] Programming arm trustzone architecture on the xilinx zynq-7000 all programmable soc. https://www.xilinx.com/support/documentation/user_guides/ug1019-zynq-trustzone.pdf.

[17] Programming arm trustzone architecture on the xilinx zynq-7000 all programmable soc user guide. https://www.xilinx.com/support/documentation/user_guides/ug1019-zynq-trustzone.pdf.

[18] Raspberry pi 3 model b. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

[19] Repositorio github rtklib. https://github.com/tomojitakasu/RTKLIB_bin/tree/rtklib_2.4.3.

[20] Royal observatory of belgium gnss research group. http://www.gnss.be/.

[21] Rtklib: An open source program package for gnss positioning. http://www.rtklib.com/.

[22] The tangle whitepaper. https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf.

[23] U-blox evk-m8f. https://www.u-blox.com/en/product/evk-8evk-m8.

[24] U-center, gnss evaluation software for windows. https://www.u-blox.com/en/product/u-center.

[25] Vivado design suite. https://www.xilinx.com/products/design-tools/vivado.html.

[26] Vivado trustzone tutorial. https://perso.univ-st-etienne.fr/bl16388h/VIVADO_TrustZone_tutorial.pdf.

[27] Zedboard development kit. http://zedboard.org/product/zedboard.

[28] ARM. Building a secure system using trustzone technology. http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf, 2008.

[29] Roger Wattenhofer Christian Decker. Information propagation in the bitcoin network. https://ieeexplore.ieee.org/document/6688704, 2013.

[30] Pascale Defraigne. Gnss time transfer. https://www.bipm.org/ws/CCTF/TAI_TRAINING/Allowed/Fundamentals/Training-2012-GNSS-Defraigne.pdf, 2012.

[31] Timothy C. May. The cyphernomicon. http://www.kreps.org/hackers/overheads/11cyphernervs.pdf, 1994.

[32] William Mougayar. *The Business Blockchain: Promise, Practice, and Application of the Next Internet Technology.*

[33] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, 2008.

[34] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. https://fc15.ifca.ai/preproceedings/paper_30.pdf.