UNIVERSIDAD NACIONAL DE LA PLATA
Facultad de Informática

Utilizando Software Libre para un servicio de Sellado Digital de Tiempo

Tesis presentada para optar al título de Licenciado en Informática

Alejandro Javier Sabolansky

Director de tesis: Lic. F. Javier Diaz
Co-director: Lic. Paula Venosa

Lugar de trabajo: Laboratorio de Investigación en Nuevas Tecnologías Informáticas

La Plata, 22 de septiembre de 2010.

Agradecimientos

Con la presentación de esta tesis se termina una etapa muy importante para mi, que comenzó hace varios años cuando decidí ingresar al mundo de la Informática y comencé a transitar por los pasillos de esta facultad.

Para empezar, quiero agradecer a mis padres y a mi hermana por el apoyo, la comprensión y la contención en todo este tiempo.

A mi director Javier Díaz, por darme la oportunidad de crecer en mi breve carrera y guiarme en el desarrollo de esta tesis.

A Paula Venosa, mi co-directora, que siempre estuvo en forma incondicional para guiarme, motivarme y aconsejarme a lo largo de todo este año de trabajo. Gracias a su empuje constante, estoy escribiendo estas líneas.

A mis compañeros de trabajo, que siempre estuvieron para darme un consejo, una sugerencia o una mano cada vez que lo necesité para poder concluir con la tesis y la presentación de la misma. Gracias por su ayuda desinteresada.

Por último, quiero agradecer a mis amigos de toda la vida por estar siempre, a mis amigos de la facultad por todos los momentos compartidos y en especial al grupo de amigos que conocí en el Laboratorio, cuyo apoyo en esta etapa fue fundamental.

A todos ellos, muchas gracias!

Alejandro Sabolansky

Índice general

Índice de figuras

Índice de cuadros

Durante el desarrollo de este trabajo, se realizó un paper titulado ***"Importancia de contar con un servicio de sellado digital de tiempo en una PKI"****. Dicho trabajo fue aceptado y presentado en formato de póster en el marco de la WICC 2010, XII Workshop de Investigadores en Ciencias de la Computación en la ciudad de El Calafate, Argentina, durante los días 5 y 6 de Mayo de 2010. Los autores de este trabajo fueron el Lic. Javier Díaz, el Lic. Nicolás Macia, la Mg. Lía Molinari, la Lic. Paula Venosa y Alejandro Sabolansky.*

Capítulo 1

Objetivos

El objetivo del presente trabajo es implementar una infraestructura de sellado de tiempo utilizando software libre, que cumpla con los estándares tecnológicos existentes, teniendo en cuenta los requisitos necesarios para brindar un servicio 7x24. Este servicio es esencial para ser integrado con sistemas de firma digital, en los cuales es fundamental tener certeza del momento exacto en que se realiza cada operación.

1.1. Motivación

El tiempo es una magnitud que afecta a todas las actividades humanas y es un componente esencial en todos los procesos, donde registrar el momento exacto en que se suceden los acontecimientos es fundamental. Una aplicación que utiliza firma digital sobre una infraestructura PKI exige que la medida de tiempo usada sea precisa y acordada.

El sellado de tiempo (Timestamping) es un mecanismo que permite demostrar que una serie de datos han existido y no han sido alterados desde un instante específico en el tiempo.

El uso de un sistema de sellado de tiempo aparece como indispensable para mantener la validez de los documentos a lo largo de los años.

1.2. Pasos a seguir

- Analizar y estudiar la normativa vigente.
- Analizar los protocolos y las distintas alternativas para implementar cada uno de los componentes de la solución.
- Seleccionar los componentes más adecuados.
- Implementar el prototipo del servicio de sellado de tiempo.

1.3. Introducción

Al momento de visualizar un documento digital surgen básicamente dos interrogantes:

- ¿Quién es el autor de este documento? ¿Quién autorizó su publicación?
- ¿Cuándo fue creado o modificado por última vez dicho documento?

En ambos casos, la pregunta es específicamente sobre ese documento y no otro. Una respuesta al primer planteo permite conocer quién y qué: Quién aprobó exactamente qué en dicho documento. La respuesta a la segunda de las preguntas planteadas permite saber cuándo y qué: Desde cuándo el contenido de ese documento existe.

Las preguntas planteadas ameritan analizar diferentes alternativas. Una alternativa para responder la primera cuestión es la firma digital, mientras que una alternativa para responder la segunda es el sistema de sellado digital de tiempo. En este marco, debe haber un procedimiento con el cuál un autor de un documento pueda firmar un conjunto de bytes que actúan como firma. Por otra parte, debe haber un mecanismo de verificación mediante el cual cualquier usuario puede chequear un documento y la firma adjunta para que, con garantía razonable, se pueda asegurar que la misma responde a las preguntas quién y qué o cuándo y qué.

La firma digital es un mecanismo orientado a garantizar la identidad del emisor de la información, la integridad y confiabilidad de la información y el no repudio tanto del emisor como del receptor.

Esta es la forma que garantiza conocer quién ha hecho qué. Pero hay un parámetro importante que la firma digital no abarca; es el instante de tiempo en que ha sucedido ese determinado suceso. Esta falencia es la que genera el surgimiento de los mecanismos de sellado digital de tiempo.

Una cuestión interesante con respecto al sellado digital de tiempo, es que el documento que está siendo sellado no debe ser enviado a ninguna entidad ni persona para poder crear el sello de tiempo. La entidad que genera el documento, calcula un valor de resumen que se corresponde en forma unívoca con el documento y envía el mismo al servicio de sellado digital de tiempo. El documento solamente es necesario para poder realizar la posterior verificación del sello de tiempo. Esto es muy importante por diversas razones, entre las cuales se pueden destacar la protección de la información por un tema de patentes o por cuestiones de seguridad y privacidad.

A lo largo del presente trabajo se profundizará sobre estos planteos iniciales para poder comprender la temática que nos ocupa, concluyendo el trabajo con la puesta en marcha de una autoridad de sellado de tiempo digital utilizando software libre en todos sus componentes involucrados.

1.4. Estructura organizativa del trabajo

La presente tesis de grado contiene 10 capítulos, que contienen la parte de investigación y la parte de implementación del prototipo propuesto.

El capítulo 2 contiene una breve introducción a la criptografía y sus conceptos asociados, haciendo mención a los métodos históricos para ayudar a formar una noción de la complejidad de los algoritmos que se explicarán posteriormente. Más adelante, se ahonda en la criptografía simétrica, en la criptografía asimétrica y por último se presenta el concepto de hashing o resumen.

El capítulo 3 contiene una explicación sobre firma digital y los temas más importantes asociados a este concepto. En primer término, se presenta una introducción de

firma digital y luego se exponen los distintos componentes de una Infraestructura de Clave Pública. Luego, se presenta el concepto de certificado digital y las extensiones asociadas. Por último, sobre la parte final de este capítulo, se hace un raconto de los principales campos de aplicación de la firma digital en el mundo actual.

El capítulo 4, comienza con una breve descripción del concepto de tiempo como magnitud física, luego explica el concepto de escala de tiempo, se presentan diversas escalas de tiempo y por último se expone NTP, un protocolo que permite la sincronización de relojes en sistemas informáticos.

El capítulo 5, presenta la normativa existente relacionada con el sellado de tiempo, dictada por diversos organismos internacionales entre los que se incluyen ANSI, ISO y la IETF. A continuación, se explica técnicamente el funcionamiento del servicio de sellado de tiempo, con sus fases y roles participantes. Una vez desarrollados estos conceptos, se presentan temas más avanzados que incluyen sellado a largo plazo, aspectos de seguridad del servicio de sellado de tiempo y el desarrollo de políticas y procedimientos. El capítulo finaliza analizando las utilidades más importantes del sellado de tiempo y el grado de evolución del servicio en algunos países del mundo.

El capítulo 6 explica los diversos esquemas de sellado de tiempo que han sido propuestos, incluyendo algunos modelos teóricos que han sido planteados por diversos académicos relacionados con el tema.

El capítulo 7 desarrolla los diversos mecanismos de transporte planteados por la RFC 3161, haciendo hincapié en el mecanismo que será utilizado en el prototipo.

El capítulo 8 contiene las decisiones de diseño y la selección de componentes que van a formar parte de la infraestructura prototípica de la Autoridad de Sellado de Tiempo. Luego de presentar diversos criterios de selección de componentes y explicar las características principales de los potenciales componentes a utilizar, se realizará una elección justificada de los mismos.

El capítulo 9 comprende la explicación de las diversas tareas de implementación realizadas junto con las pruebas de funcionamiento del prototipo. Entre las configuraciones realizadas se encuentran la instalación de los servidores de base de datos, de los servidores web, los servidores de tiempo, la Autoridad de Certificación para la emisión de certificados y el desarrollo de una interfaz web que permite la interacción con el servicio.

Por último, el capítulo 10 contiene los agregados de seguridad y monitoreo que se han integrado con el servicio de sellado digital de tiempo para lograr brindar un servicio que esté accesible 7X24.

Capítulo 2

Criptografía

Los documentos que se generan en forma electrónica tienen asociados tres conceptos importantes que se deben resguardar: la confidencialidad, la integridad y la disponibilidad.

La confidencialidad asegura que la información sólo sea accesible por los individuos autorizados.

La integridad garantiza que la información no ha sido alterada ni durante el proceso de transmisión ni en el propio equipo donde se originó la misma.

La disponibilidad de la información garantiza que los usuarios autorizados tengan acceso a la información y a los recursos relacionados con la misma, toda vez que lo requieran.

Para poder cumplir con estos aspectos, que se conocen como los objetivos principales de la seguridad de la información, se utiliza una técnica que se conoce como criptografía.

A lo largo de este capítulo, se hará una breve introducción a la criptografía y sus conceptos asociados, haciendo mención a los métodos históricos para ayudar a formar una noción de la complejidad de los algoritmos que se explicarán posteriormente. Más adelante, se ahondará en la criptografía simétrica, en la criptografía asimétrica y por último se presentará el concepto de hashing o resumen. En todas las secciones y subsecciones se explicarán los temas presentando implementaciones concretas de algoritmos, cuya comprensión será transcendental al momento de implementar el prototipo planteado en este trabajo.

2.1. Introducción a la criptografía

La palabra criptografía[1] proviene del griego *kriptos* que significa escondido y *graphos* que significa escritura y según el diccionario se define como "el arte de escribir con clave secreta o de un modo enigmático". La criptografía es la ciencia de utilizar las matemáticas para encriptar y desencriptar datos. La criptografía permite almacenar información sensible o transmitirla a través de medios inseguros, como es Internet, de forma tal que no puede ser leída por nadie excepto por las entidades autorizadas.

Los servicios de cifrado son la base para muchas implementaciones de seguridad y se utilizan para garantizar la protección de los datos cuando los mismos podrían estas expuestos a partes que no son de confianza. La historia de la criptografía se inicia en los círculos diplomáticos hace miles de años atrás.

Un algoritmo criptográfico, es una función matemática utilizada en el proceso de encripción y desencripción. El mismo funciona en combinación de una clave, que puede ser una palabra, un número o una frase, por ejemplo, para encriptar el texto plano. El mismo texto plano se transforma en diferente texto cifrado cuando se utilizan diferentes claves para realizar el proceso. La seguridad de los datos encriptados es exclusivamente dependiente de dos factores: la fuerza del algoritmo criptográfico utilizado y el secreto de la clave.

La conjunción de un algoritmo criptográfico junto con todas las posibles claves y todos los protocolos necesarios para hacer funcionar el algoritmo, conforman un criptosistema.

Dentro de la criptografía[2] existe una clara división entre dos tipos de cifrado dependiendo del tipo de claves que se utilizan para el procesamiento de los mensajes.

Varios métodos de cifrado y dispositivos físicos han sido utilizados para encriptar y desencriptar textos:

- Uno de los métodos más antiguos fue el *scytale* de la antigua Grecia, una barra utilizada por los espartanos como una ayuda para el cifrado de transposición. El emisor y el receptor tenían barras idénticas que permitían envolver el mensaje y leer con claridad.

- El cifrado César es uno de los primeros métodos de cifrado conocidos históricamente. El emperador Julio César lo usó para enviar órdenes a sus generales en los campos de batalla. Consistía en escribir el mensaje con un alfabeto que estaba formado por las letras del alfabeto latino normal desplazadas tres posiciones a la derecha. El receptor del mensaje conocía la clave secreta de éste (es decir, que estaba escrito con un alfabeto desplazado tres posiciones a la derecha), y podía descifrarlo fácilmente haciendo el desplazamiento inverso con cada letra del mensaje. Pero para el resto de la gente que pudiese accidentalmente llegar a ver el mensaje, el texto carecía de sentido.

- El cifrado Vigenère es un cifrado basado en diferentes series de caracteres o letras del cifrado César formando estos caracteres una tabla, llamada tabla de Vigenère, que se usa como clave. El cifrado de Vigenère es un cifrado polialfabético y de sustitución.

- Thomas Jefferson, el tercer presidente de los Estados Unidos, inventó un sistema de cifrado que se cree que utilizó cuando fue secretario de estado, entre 1790 y 1793.

- Arthur Scherbius inventó en 1918 un dispositivo de codificación electromecánico llamado Enigma que vendió a Alemania. Este dispositivo sirvió como modelo para las máquinas que utilizaron todos los grandes participantes de la Segunda Guerra Mundial. Se estima que si 1000 criptoanalistas probaran cuatro claves por minuto, todo el día todos los días, tomaría 1.800.000.000 años probarlas a todas. Alemania sabía que sus mensajes cifrados podían ser interceptados por los aliados, pero nunca pensó que podrían descifrarlos.

- También durante la Segunda Guerra Mundial, Japón descifró todos los códigos que inventaron los americanos. Era necesario un sistema de codificación más elaborado

y la respuesta surgió en la forma de traductores de código navajo. No sólo no existían palabras en el idioma navajo para los términos militares, sino que el lenguaje no posee escritura. Además, menos de 30 personas fuera de las reservas navajos podían hablarlo y ninguno de ellos era japonés. Hacia el final de la guerra, más de 400 indios navajo trabajaban como traductores de código.

2.2. Métodos de cifrado

Cada uno de estos métodos de cifrado enunciados, utiliza un algoritmo específico, llamado cifrador, para cifrar y descifrar mensajes. Un cifrador consiste en una serie de pasos bien definidos, los cuales pueden ejecutarse como un procedimiento para cifrar y descifrar mensajes.

Existen diferentes métodos para crear texto cifrado:

- Método de transposición.
- Método de sustitución.
- Método de Vernam.

2.2.1. Método de transposición

En los cifrados por transposición, las letras no se reemplazan, sino que son reordenadas. Un ejemplo de este tipo de cifrado es tomar el mensaje *FLANK EAST ATTACK AT DAWN* y luego transponerlo como *NWAD TAKCATTA TSAE KNALF*. En este ejemplo, la clave es revertir las letras.

Otro ejemplo de un cifrado por transposición es conocido como Cifrado Rail Fence. En esta transposición, las palabras son deletreadas como si se tratara de los rieles de una valla, es decir, algunos por delante y otros por detrás de varias líneas paralelas. Por ejemplo, un cifrado Rail Fence que utiliza "tres" como clave, especifica que se requieren tres líneas para crear el código cifrado. Para leer el mensaje, debe hacerse en forma diagonal, de arriba abajo, siguiendo las líneas de la valla.

F...K...T...A...T...N.
.L.N.E.S.A.T.C.A.D.W..
..A...A...T...K...A...

Los algoritmos de cifrado modernos, tales como DES (Data Encryption Standard) y 3DES (Triple Data Encryption Standard) todavía utilizan transposición como parte del algoritmo.

2.2.2. Método de sustitución

Los cifradores por sustitución reemplazan una letra por otra. En su forma original, los cifrados por sustitución retienen la frecuencia de letras del mensaje original. El cifrado César era un cifrado por sustitución simple. Cada día se utilizaba una clave

diferente para ajustar el alfabeto. Por ejemplo, si la clave del día era *3*, la letra *A* era desplazada tres espacios a la derecha, resultando en un mensaje codificado que utilizaba la letra *D* en lugar de la letra *A*. Como el mensaje completo se basaba en el mismo desplazamiento clave, el Cifrado César es conocido como un cifrado de sustitución monoalfabético. Es también relativamente fácil de romper. Por este motivo, se inventaron los cifrados polialfabéticos, tales como el Cifrado Vigènere. El método fue descripto originalmente por Giovan Battista Bellaso en 1553, pero el esquema fue atribuido más tarde en forma errónea al diplomático y criptógrafo francés Blaise de Vigènere. El Cifrado Vigènere se basa en el Cifrado César, excepto en que cifra el texto utilizando diferentes desplazamientos polialfabéticos de clave para cada letra del texto plano. Los diferentes desplazamientos son identificados utilizando una clave compartida entre el autor y el receptor. El mensaje en texto plano puede ser cifrado y descifrado utilizando la Tabla de Cifrado Vigènere.

2.2.3. Método de Vernam

Gilbert Vernam fue un ingeniero de AT&T Bell Labs que, en 1971, inventó y patentó el cifrador de flujo y luego co-inventó la "tres". Vernam propuso un cifrador de teletipo, el cual registraba en una cinta de papel una clave preparada que consistía en una secuencia de números no repetidos y de longitud arbitraria. Esta clave era luego combinada letra por letra con el mensaje en texto plano para producir el criptograma. Para descifrar el criptograma, se combinaba letra por letra la misma cinta de papel, produciendo el texto plano original. Cada cinta se utilizaba una sola vez, de allí el término "libreta de un solo uso". Mientras la cinta clave no se repitiera ni fuera reutilizada, este tipo de cifrado era inmune a los ataques de criptoanálisis, ya que los criptogramas disponibles no revelan el patrón de la clave. La utilización de libretas de un solo uso en el mundo real presenta algunas dificultades inherentes. Una de ellas es el desafío de crear datos aleatorios. Debido a que poseen una base matemática, las computadoras son incapaces de crear datos realmente aleatorios. Además, si la clave es utilizada más de una vez, se vuelve fácil de descubrir. RC4 es un ejemplo de este tipo de cifrado ampliamente utilizado en Internet. Nuevamente, debido a que las claves son generadas por una computadora, no son verdaderamente aleatorias. Además de estos problemas, la distribución de la clave también es un desafío para este tipo de cifrado.

2.3. Criptografía simétrica

La criptografía simétrica es un método criptográfico en el cuál se usa una misma clave para cifrar y descifrar los mensajes. Las dos partes que se comunican han de ponerse de acuerdo de antemano sobre la clave a utilizar. Una vez que ambas partes de la comunicación tienen acceso a la clave, el remitente lo cifra utilizando la clave y el destinatario, una vez que lo recibe, obtiene el mensaje original aplicando la clave sobre el mensaje recibido.

Las técnicas más utilizadas de criptografía de cifrado simétrico son los cifrados de bloque y los cifrados de flujo.

2.3.1. Cifrado por bloques

El cifrado por bloques (*block cipher*) transforma un bloque de texto plano de longitud fija en un bloque criptográfico común de 64 o 128 bits. El tamaño del bloque define qué cantidad de datos puede cifrarse por vez. Actualmente, el tamaño del bloque, también conocido como longitud fija, es en general de 64 o 128 bits. La longitud de la clave se refiere a la clave de cifrado utilizada. Este criptograma es descifrado aplicando la transformación inversa del bloque criptográfico, utilizando la misma clave secreta. El cifrado por bloques en general resulta en una salida de datos más larga que los datos de entrada, debido a que el criptograma debe ser un múltiplo del tamaño de los bloques. Por ejemplo, DES cifra bloques en porciones de 64 bits, utilizando una clave de 56 bits. Para lograr esto, el algoritmo de bloques toma una porción de datos por vez, por ejemplo porciones de 8 bytes, hasta que se rellena el bloque completo. Si hay menos datos que los necesarios para completar un bloque, el algoritmo agrega datos artificiales (blancos) hasta que los 64 bits son utilizados. El cifrado por bloques incluye DES con bloques de 64 bits, AES con bloques de 128 bits y RSA con bloques de tamaño variable.

2.3.2. Cifrado de flujo

A diferencia del cifrado por bloques, el cifrado de flujo (*stream cipher*) codifica el texto plano de a un byte o un bit por vez. Los cifrados de flujo pueden verse como un cifrado de bloque con un tamaño de bloque de un bit. Con el cifrado de flujo, la transformación de estas unidades más pequeñas de texto plano es variable, dependiendo de dónde se encuentren durante el proceso de cifrado. El cifrado de flujo puede ser mucho más rápido que el cifrado por bloques y en general no incrementa el tamaño de los mensajes, ya que pueden cifrar un número arbitrario de bits. El cifrado Vigènere es un ejemplo del cifrado de flujo. Este cifrado es periódico, debido a que su clave es de longitud infinita y la misma se repite si es más corta que el mensaje. El cifrado de flujo incluye A5, el cual es utilizado para cifrar las comunicaciones de teléfonos celulares GSM y el cifrado RC4. DES también puede ser utilizado en modo de cifrado de flujo.

2.3.3. Algoritmo DES

El Data Encryption Standard (DES)[3] es un algoritmo de cifrado simétrico, utilizado normalmente en modo de cifrado por bloques. El algoritmo DES es en esencia una secuencia de permutaciones y sustituciones de bits de datos, combinadas con una clave de cifrado. Se utiliza el mismo algoritmo y la misma clave tanto para el cifrado como el descifrado. DES tiene una longitud de clave fija. La clave tiene una longitud de 64 bits, pero sólo se utilizan 56 bits para el cifrado. Los 8 bits restantes son utilizados para la paridad. El bit menos significativo de cada byte de la clave es utilizado para indicar la paridad impar. Una clave DES siempre tiene una longitud de 56 bits. Cuando se utiliza DES con una clave débil de 40 bits, la clave de cifrado contiene 40 bits secretos y 16 bits conocidos, lo cual conforma una clave de 56 bits. En este caso, la fortaleza de la clave DES es de 40 bits.

Debido a la breve longitud de sus claves, DES es considerado un buen protocolo para proteger datos por un período corto de tiempo, sin embargo 3DES es una mejor opción para proteger datos debido a que la longitud de la clave es mayor, y a diferencia

de DES, no se han encontrado ataques de fuerza bruta con resultado positivo.

2.3.4. 3DES

Con los avances en el poder de procesamiento de las computadoras, las claves DES originales de 56 bits se volvieron demasiado cortas para soportar ataques realizados con tecnología de mediano presupuesto. Una forma de aumentar la longitud efectiva de la clave DES, sin modificar el algoritmo bien analizado, consiste en utilizar repetidas veces el mismo algoritmo con diferentes claves. La técnica de aplicar DES tres veces seguidas a un mismo bloque de texto plano es conocida como 3DES. En la actualidad, los ataques por fuerza bruta sobre 3DES son considerados impracticables debido a que los algoritmos básicos han sido bien analizados durante sus más de 35 años de utilización. Por lo tanto, es considerado muy confiable.

3DES utiliza un método llamado 3DES-Encrypt-Decrypt-Encrypt (3DES-EDE) para cifrar texto plano. Primero, el mensaje es cifrado utilizando la primera clave de 56 bits, llamada K1. Luego, los datos se descifran utilizando la segunda clave de 56 bits, llamada K2. Finalmente, los datos son nuevamente cifrados con la tercera clave de 56 bits, llamada K3. El procedimiento 3DES-EDE es mucho más efectivo en el aumento de la seguridad que el simple cifrado de los datos tres veces con tres claves diferentes. Cifrando datos tres veces consecutivas utilizando claves diferentes de 56 bits equivale a una fortaleza de clave de 58 bits. El procedimiento 3DES-EDE, por otro lado, provee un cifrado con una longitud efectiva de clave de 168 bits. Si las claves K1 y K3 son iguales, como sucede en algunas implementaciones, se obtiene un cifrado menos seguro de 112 bits. Para descifrar el mensaje, debe utilizarse el proceso inverso a 3DES-EDE. Primero, el criptograma se descifra utilizando la clave K3. Luego, los datos son cifrados utilizando la clave K2. Finalmente, los datos vuelven a descifrarse utilizando la clave K1. Aunque 3DES es muy seguro, también consume recursos en forma intensiva. Por este motivo, se desarrolló el algoritmo de cifrado AES, el cual ha sido probado tan seguro como 3DES, pero con resultados mucho más veloces.

2.3.5. Algoritmo AES

Durante algunos años, se había pensado que DES llegaría eventualmente al final de su utilidad. En 1997 fue anunciada la iniciativa AES[4][5] y se invitó al público a proponer esquemas de cifrado para reemplazar a DES. Luego de un proceso de estandarización de 5 años, durante los cuales se presentaron y evaluaron 15 diseños diferentes, el Instituto Nacional de Estándares y Tecnología de los Estados Unidos (NIST) seleccionó el cifrado por bloques de Rijndael para el algoritmo AES. El cifrado Rijndael, desarrollado por Joan Daemen y Vincent Rijmen, posee una longitud de bloque y longitud de clave variables. Rijndael es un cifrado por bloques iterativo, lo que significa que el bloque de entrada inicial y la clave de cifrado atraviesan múltiples ciclos de transformación antes de generar los datos de salida. El algoritmo puede operar sobre bloques de tamaños variables, utilizando claves de diferente longitud. Pueden utilizarse claves de 128 bits, 192 bits o 256 bits, para cifrar bloques de datos de 128, 182 o 256 bits de longitud y es posible utilizar cualquiera de las nueve combinaciones de bloques y claves. La implementación Rijndael aceptada para AES contiene sólo algunas capacidades del algoritmo Rijndael. El algoritmo fue escrito de forma tal que la longitud de los bloques

o de la clave puedan ser fácilmente extendidos en múltiplos de 32 bits y el sistema está diseñado específicamente para su implementación por hardware o por software en un amplio rango de procesadores. El algoritmo AES ha sido analizado extensivamente y ahora es utilizado en todo el mundo. Aunque no ha sido probado con el uso cotidiano en igual grado que 3DES, AES con cifrado Rijndael es el algoritmo más eficiente. Puede ser utilizado en ambientes de baja latencia y gran volumen de transferencia, especialmente cuando 3DES no puede procesar los requerimientos de latencia o transferencia. Se espera que aumente la confianza en AES con el transcurso del tiempo, a medida que se intenten más ataques en su contra. AES fue seleccionado para reemplazar a DES por diferentes razones. La longitud de clave de AES la vuelve mucho más segura que DES; AES se ejecuta con mayor velocidad que 3DES sobre hardware de similares características; AES es más eficiente que DES y 3DES, usualmente por un factor de cinco cuando es comparado con DES; AES es más adecuado en ambientes de baja latencia y alta transferencia, especialmente si se utiliza sólo cifrado por software. A pesar de estas ventajas, AES es un algoritmo relativamente joven. La regla de oro de la criptografía establece que un algoritmo maduro es siempre más confiable. Por lo tanto, 3DES es la opción más confiable en términos de fortaleza, porque ha sido probado y analizado por 35 años.

2.4. Criptografía asimétrica

Los algoritmos asimétricos, también conocidos como algoritmos de clave pública, están diseñados de forma tal que la clave utilizada para cifrar los datos sea diferente a la utilizada para descifrarlos. La clave de descifrado no puede ser calculada en un tiempo razonable a partir de la clave de cifrado y viceversa. Los algoritmos asimétricos utilizan dos claves: una clave pública y una clave privada. Ambas pueden utilizarse en el proceso de cifrado, pero se requiere la clave complementaria correspondiente para su descifrado. Por ejemplo, si se utiliza una clave pública para cifrar los datos, la clave privada correspondiente los descifra. La operación inversa también es válida: si se cifran los datos con la clave privada, debe utilizarse la clave pública correspondiente para descifrarlos.

Este proceso permite que los algoritmos asimétricos proporcionen autenticación, integridad y confidencialidad.

2.4.1. Algoritmo DSA

En 1994, el instituto NIST de los Estados Unidos seleccionó a DSA como el Estándar de Firma Digital (DSS - Digital Signature Standard)[6]. DSA está basado en el problema del logaritmo discreto y sólo puede proveer firmas digitales. Sin embargo, DSA ha recibido algunas críticas. Quienes lo critican aducen que DSA no posee la flexibilidad de RSA. La verificación de firmas es demasiado lenta y el proceso mediante el cual el NIST seleccionó a DSA fue demasiado secreto y arbitrario. En respuesta a estas críticas, DSS ahora incorpora dos posibles algoritmos adicionales: Criptografía de Clave Pública Reversible Utilizando Firma Digital (la cual utiliza RSA) y el Algoritmo de Firma Digital de Curva Elíptica (ECDSA - Elliptic Curve Digital Signature Algorithm). La generación de firmas DSA es más veloz que la verificación de firmas DSA. Por otro lado,

la verificación de firmas en RSA es más veloz que la generación de firmas.

2.4.2. Algoritmo RSA

RSA es uno de los algoritmos asimétricos más populares. Ron Rivest, Adi Shamir y Len Adleman inventaron el algoritmo RSA en 1977[7]. Se trata de un algoritmo patentado de clave pública. Su patente expiró en septiembre del año 2000 y el algoritmo es actualmente de dominio público. De todos los algoritmos de clave pública que fueron propuestos a través de los años, RSA es el más sencillo de comprender e implementar. El algoritmo RSA es muy flexible porque posee una clave de longitud variable, por lo que la clave puede ser acortada para acelerar el procesamiento. A cambio, mientras más corta es la clave, menos seguro es el algoritmo. Las claves RSA tienen por lo general entre 512 y 2048 bits de longitud. RSA ha soportado años de criptoanálisis intensivo. Aunque la seguridad de RSA nunca ha sido probada o refutada, esto sugiere cierto nivel de confianza en el algoritmo. La seguridad de RSA se basa en la dificultad de factorizar números muy grandes. En caso de descubrirse un método sencillo para factorizar estos números grandes, la efectividad de RSA se vería destruida. El algoritmo RSA se basa en una clave pública y en una clave privada. La clave pública puede ser distribuida, pero la clave privada debe ser mantenida en secreto. No es posible determinar la clave privada a partir de la clave pública, utilizando ningún algoritmo computable y viceversa. Las claves RSA son de largo plazo y en general son cambiadas o renovadas luego de algunos meses o incluso años. Actualmente es el método más común de generación de firmas y es ampliamente utilizado en los sistemas de comercio electrónico y protocolos de Internet. RSA implementado por hardware es unas cien veces más lento que DES, e implementado por software es unas mil veces más lento que DES. Este problema de rendimiento es la razón principal por la que RSA sólo es utilizada en general para proteger pequeñas cantidades de datos. RSA es utilizado principalmente para asegurar la confidencialidad de los datos mediante el cifrado y para la autenticación o no repudio de los datos, o ambos, mediante la generación de firmas digitales.

2.5. Hashing

Una función de *hash* es un procedimiento determinístico que toma un bloque arbitrario de datos y retorna una secuencia de caracteres de una longitud fija, lo que se conoce como valor de hash. Los datos que van a ser codificados generalmente se los llama "mensajes" y el valor de hash resultante es conocido como "digesto del mensaje" o simplemente "digesto". Un cambio intencional o accidental en el mensaje va a generar una modificación en digesto. Una función de hash debe poder resistir a los conocidos ataques de criptoanálisis. Para ello, la función de hash debería cumplir con estas propiedades:

1. **Resistencia de preimagen:** Dado un hash h debe ser difícil encontrar un mensaje m tal que $h = \text{hash}(m)$. El concepto se relaciona con la función de una vía. Las funciones que no cumplen con esta propiedad son vulnerables a ataques de preimagen.

2. **Segunda resistencia de preimagen:** Dado un dato de entrada *m1*, debe ser difícil encontrar otro dato de entrada, *m2*, distinto de m1, tal que hash(*m1*) = hash(*m2*). Esta propiedad se relaciona con la resistencia a la colisión.

3. **Resistencia a la colisión :** Debe ser difícil encontrar dos mensajes *m1* y *m2* tal que hash(*m1*) = hash(*m2*). Ese par es conocido como colisión de hash.

Estas propiedades implican que un atacante no puede modificar el dato de entrada sin cambiar el hash resultante. Por lo tanto, si dos datos tienen el mismo digesto, uno puede asegurar que son idénticos.

2.5.1. MD5

MD5[8] es una función de resumen criptográfica ampliamente utilizada de 128 bits; es un estándar de Internet, descripto en la RFC 1312. MD5 es utilizado en una gran cantidad de aplicaciones de seguridad y también es muy conocido para el chequeo en la integridad de los archivos. Sin embargo, se ha demostrado que MD5 no es resistente a las colisiones[9], por lo tanto no debería utilizarse para aplicaciones como certificados SSL que confían en dicha propiedad. Un resumen o hash MD5 típicamente es expresado en un número en formato hexadecimal de 32 bits.

MD5 es esencialmente una secuencia compleja de operaciones binarias simples, tales como OR Exclusivo (XOR) y rotaciones, que se ejecutan sobre los datos y producen un digesto del mensaje de 128 bits. El algoritmo principal se basa en una función de compresión, la cual opera sobre bloques. La entrada es un bloque de datos más un resultado de bloques previos. Los bloques de 512 bits se dividen en 16 sub-bloques de 32 bits. Estos bloques son luego reordenados con operaciones simples en un bucle principal, el cual consiste en cuatro iteraciones. La salida del algoritmo es un conjunto de cuatro bloques de 32 bits que se concatenan para formar un único valor de hash de 128 bits.

MD5 fue diseñado por Ron Rivest en 1991 en reemplazo de una antigua función de hash, MD4. En el año 1996, una falla fue encontrada en el diseño de MD5. Si bien no fue claramente una debilidad fatal, los expertos comenzaron a recomendar el uso de otros algoritmos, como SHA-1. En el año 2004, fallas más serias fueron descubiertas mientras que en el 2007 un grupo de investigadores describió como crear un par de archivos que comparten el mismo resumen de MD5. En el año 2008, otro grupo de investigadores utilizó dicha técnica para crear certificados de SSL fraudulentos.

El gobierno de Estados Unidos considera al protocolo criptográficamente roto y recomienda el cese de utilización del mismo.

2.5.2. SHA

El Instituto Nacional de Estándares y Tecnología (NIST) de los Estados Unidos desarrolló SHA (Secure Hash Algorithm), el algoritmo especificado en SHS (Secure Hash Standard). SHA-1, publicado en 1994, corrige una falla no publicada de SHA. Su diseño es muy similar al de las funciones de hash MD4 y MD5 desarrolladas por Ron Rivest. El algoritmo SHA-1[10] toma un mensaje con menos de 264 bits de longitud y produce un digesto de 160 bits. El algoritmo es apenas más lento que MD5, pero al

generar un digesto más largo, es más seguro contra ataques de colisión por fuerza bruta y ataques de inversión. El NIST publicó cuatro funciones de hash adicionales para la familia SHA, cada uno con digestos más largos:

- SHA-224 (224 bits)
- SHA-256 (256 bits)
- SHA-384 (384 bits)
- SHA-512 (512 bits)

Estas cuatro versiones son conocidas en forma conjunta como SHA-2, aunque el término SHA-2 no ha sido estandarizado. SHA-1, SHA-224, SHA-256, SHA-384 y SHA-512 son los algoritmos seguros de hash requeridos por ley para su utilización en ciertas aplicaciones del gobierno de los Estados Unidos, incluyendo su uso dentro de otros algoritmos y protocolos criptográficos, para la protección de información sensible no clasificada.

Capítulo 3

Firma digital

En este capítulo se hará una explicación sobre firma digital y los temas más importantes asociados a este concepto. En primer término, se hará una introducción sobre firma digital y luego se expondrán los distintos componentes de una Infraestructura de Clave Pública conocida como PKI, por sus siglas en inglés. Luego, se presentará el concepto de certificado digital y las extensiones asociadas; la comprensión de estos temas resultará fundamental para poder poner en marcha la infraestructura de sellado de tiempo propuesta. Por último, sobre la parte final de este capítulo, se hará un raconto de los principales campos de aplicación de la firma digital en el mundo actual.

3.1. Introducción

La firma digital es un esquema matemático para demostrar la autenticidad de un mensaje digital o documento. Una firma digital válida otorga al destinatario razones para creer que el mensaje fue generado por un remitente conocido y que no fue alterado durante la transmisión del mismo. La firma digital utiliza criptografía asimétrica. La firma digital de un documento es el resultado de aplicar cierto algoritmo matemático al contenido, y a continuación aplicar un cifrado de tipo asimétrico (utilizando la clave privada del firmante) al resultado de la operación anterior. Un esquema tradicional de firma digital consiste de tres algoritmos:

- Un algoritmo de generación de claves que selecciona una clave privada al azar de un conjunto de posibles claves privadas. El algoritmo devuelve la clave privada y su correspondiente clave pública.
- Un algoritmo de firmado, el cual dado un mensaje y una clave privada, genera una firma.
- Un algoritmo de verificación de firma, el cual dado un mensaje, una clave pública y una firma, acepta o rechaza la autenticidad del mensaje.

Dos propiedades principales son necesarias para el correcto funcionamiento de este esquema. En primer lugar, una firma generada desde un mensaje determinado y una clave privada debe verificar la autenticidad de dicho mensaje utilizando la clave pública correspondiente. Por otra parte, debe ser computacionalmente imposible generar una firma válida por alguien que no posea la clave privada.

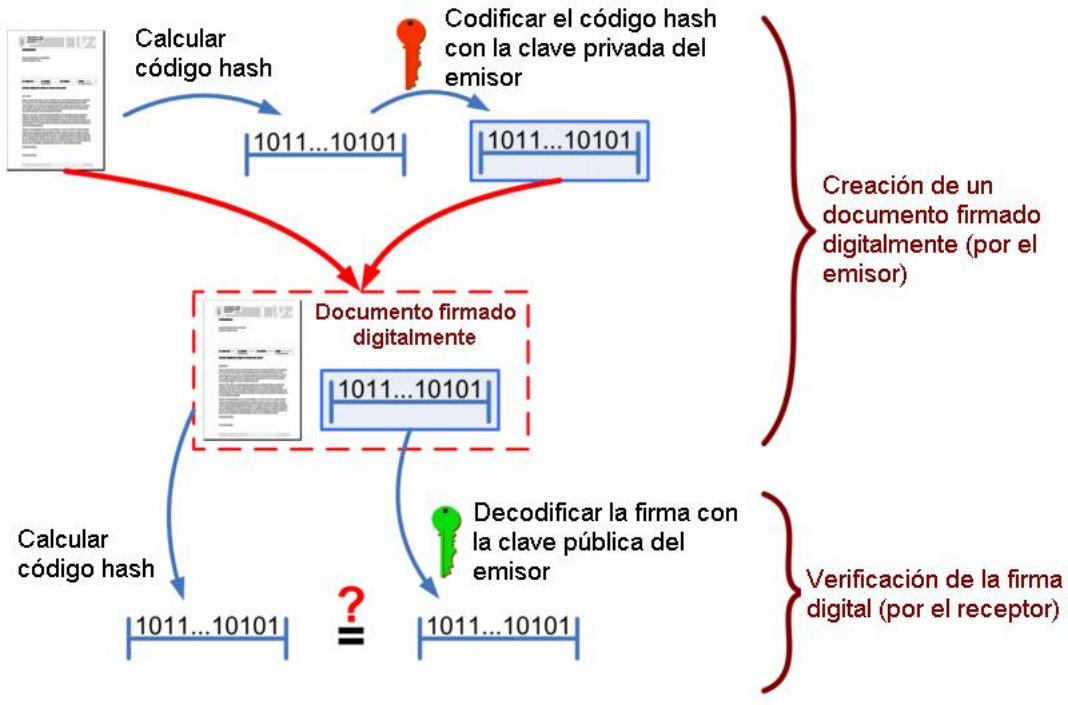

Figura 3.1: Esquema básico de firma digital

La función hash es un algoritmo matemático que permite calcular un valor resumen de los datos a ser firmados digitalmente. Funciona en una sola dirección, es decir, no es posible calcular los datos originales a partir del valor del resumen. Cuando la entrada es un documento, el resultado de la función es un valor que identifica inequívocamente al texto. Si se adjunta este valor al texto, el destinatario puede aplicar de nuevo la función y comprobar su resultado con el que ha recibido.

Las normas TS 101 733 y TS 101 903 definen los formatos técnicos de la firma electrónica. La primera se basa en el formato clásico PKCS7 y la segunda en XML-DSig, que consiste en la firma XML especificada por el consorcio W3C. Bajo estas normas se definen tres modalidades de firma:

- **Firma básica**. Incluye el resultado de operación de hash firmado con la clave privada del firmante, el certificado asociado a la clave privada del firmante e identifica los algoritmos utilizados en el proceso de firma.
- **Firma fechada**. A la firma básica se añade un sello de tiempo calculado a partir del hash del documento y firmado por una TSA (Time Stamping Authority, autoridad de sellado de tiempo en español).
- **Firma validada o firma completa**. A la firma fechada se añade información sobre la validez del certificado procedente de una consulta de CRL (Certificate Revocation List, lista de revocación de certificados en español) o de OCSP (Online Certificate Status Protocol, en español protocolo en línea de estado de certificados) realizada a la CA (Certification Authority, autoridad de certificación) pertinente.

La firma completa libera al receptor de la firma del problema de ubicar al prestador de servicios de certificación responsable de gestionar las consultas de validez del certificado

del firmante, y de determinar los procedimientos de validación disponibles.

3.2. Infraestructura PKI

Una infraestructura de clave pública PKI[11] (Public Key Infrastructure) es una combinación de componentes de hardware y software, personas, políticas y procedimientos de seguridad que permiten la ejecución con garantías de operaciones criptográficas. Dichas operaciones pueden ser el cifrado, la firma digital o el no repudio de transacciones electrónicas. Una infraestructura de clave pública (PKI) establece un vínculo entre claves públicas y sus respectivas identidades de usuario por medio de una Autoridad de Certificación (CA por sus siglas en inglés, Certification Authority). La identidad de usuario debe ser única para cada Autoridad de Certificación. Este vínculo usuario-claves se establece a través de un proceso de registro y expedición, que, dependiendo de lo establecido en las políticas y procedimientos, será realizado mediante software en una Autoridad de Certificación o bajo supervisión humana. La infraestructura PKI permite a los usuarios autenticarse frente a otros usuarios y usar la información de los certificados de identidad (por ejemplo, las claves públicas de otros usuarios) para cifrar y descifrar mensajes, firmar digitalmente información, garantizar el no repudio de un envío, y otros usos. En una operación criptográfica que use infraestructura PKI, intervienen conceptualmente como mínimo las siguientes partes:

- Un usuario que inicia la operación.
- Una serie de autoridades, que den fe de la ocurrencia de la operación y garanticen la validez de los certificados implicados en la misma (autoridad de certificación, Autoridad de registro y sistema de Sellado de tiempo).
- Un destinatario de los datos cifrados/firmados/enviados, garantizados por parte del usuario que inició la operación (puede ser él mismo).

Las operaciones criptográficas de clave pública son procesos en los que se utilizan algoritmos de cifrado conocidos y accesibles para todos. Por este motivo la seguridad que puede aportar la tecnología PKI, está fuertemente ligada a la privacidad de la llamada clave privada y los procedimientos operacionales y políticas de seguridad que se aplican.

Cabe destacar la importancia de la aplicación y cumplimiento de las políticas de seguridad en esta tecnología, puesto que ni los dispositivos más seguros ni los algoritmos de cifrado más fuertes sirven de nada si, por ejemplo, una copia de la clave privada protegida por una tarjeta inteligente criptográfica se guarda en medio inseguro sin tomar los recaudos adecuados.

3.3. Componentes de la infraestructura PKI

En esta sección se definen los componentes principales que conforman una infraestructura de clave pública PKI. Puede darse el caso de que una misma entidad cumpla las funciones de más de uno de los componentes mencionados a continuación.

3.3.1. Usuario subscriptor

Se entiende por usuario suscriptor de la PKI al usuario que voluntariamente confía y hace uso de los certificados de los que es titular. Posee (al menos) un par de claves (pública y privada) junto con un certificado asociado a su clave pública y utiliza un conjunto de aplicaciones que hacen uso de la tecnología PKI para validar firmas digitales, cifrar documentos para otros usuarios, entre otras acciones.

3.3.2. Autoridad de certificación (CA)

La autoridad de certificación CA es la entidad de confianza que da legitimidad a la relación de una clave pública con la identidad de un usuario o servicio. Se encarga de emitir y revocar certificados: expide un certificado de clave pública para cada usuario, con el que la identidad del usuario asociada a su clave pública, así como las condiciones de validez y otros atributos, son infalsificables. La CA, en forma independiente o mediante la intervención de una autoridad de registro RA, verifica la identidad del solicitante de un certificado antes de su expedición o, en caso de certificados expedidos con la condición de revocados, elimina la revocación de los certificados al comprobar dicha identidad. Los certificados son documentos que recogen ciertos datos de su titular y su clave pública. Se encuentran firmados electrónicamente por la CA, utilizando la clave privada de ésta. La CA es un tipo particular de prestador de servicios de certificación, que legitima ante terceros, que confían en sus certificados, la relación entre la identidad de un usuario y su clave pública. La confianza de los usuarios en la CA es importante para el correcto funcionamiento del servicio y justifica la filosofía de su empleo, pero no existe un procedimiento normalizado para demostrar que una CA merece dicha confianza.

3.3.3. Autoridad de registración (RA)

La autoridad de registración RA (del inglés Registration Authority) es el componente responsable de verificar la ligadura entre los certificados (concretamente, entre la clave pública del certificado) y la identidad de sus titulares. Es una parte de la PKI que mantiene las identidades de aquellos usuarios de los que las autoridades de certificación pueden expedir certificados digitales.

3.3.4. Repositorios

Los repositorios son las estructuras encargadas de almacenar la información relativa a la PKI. Los dos repositorios más importantes son el repositorio de certificados y el repositorio de listas de revocación de certificados. En una lista de revocación de certificados (o, en inglés, CRL, Certificate Revocation List) se incluyen todos aquellos certificados que por algún motivo han dejado de ser válidos antes de la fecha establecida dentro del mismo certificado.

3.4. Certificados digitales

Un certificado digital es un documento digital a través del cual una tercera parte confiable, en este caso una autoridad de certificación, garantiza la vinculación entre el sujeto o entidad y su clave pública.

El certificado es definido de acuerdo al estándar X.509. A lo largo del tiempo, el certificado X.509 ha evolucionado para ser más flexible y poderoso y puede ser utilizado para contener una gran variedad de información, mucha de la cual es opcional. El certificado X.509 está protegido por la firma digital del emisor y los usuarios pueden verificar que el contenido del mismo no fue alterado mediante la verificación del mismo. Los certificados contienen un conjunto de campos comunes, y también pueden incluir una variedad de extensiones. Existen diez campos comunes, seis de los cuales son obligatorios y cuatro son opcionales. Los campos obligatorios son: número de serie, identificador del algoritmo de la firma, nombre del emisor del certificado, período de validez, clave pública y el nombre del sujeto. Los campos opcionales son: número de versión, identificadores únicos tanto de emisor como sujeto y las extensiones. Los campos opcionales aparecen únicamente en los certificados X.509 v2 y X.509 v3.

A continuación se explica en detalle la función de cada campo en un certificado X.509.

- **Versión**: El campo de versión describe la sintaxis del certificado. Existen tres tipos de versiones de certificados. Cuando el campo es omitido, el certificado está codificado en versión 1. La versión 1 no incluye identificadores ni extensiones. La versión 2 incluye identificadores pero no incluye extensiones. En la versión 3 se incluyen ambos componentes y es la versión más utilizada actualmente.

- **Número de serie**: El número de serie es un número entero asignado por el emisor del certificado, la autoridad de certificación. Este número debe ser único para cada certificado generado. La combinación entre el número de serie y el nombre del emisor identifica inequívocamente a cualquier certificado.

- **Firma**: El campo de firma indica cuál fue el algoritmo de firma digital que fue utilizado para proteger el certificado. Un ejemplo es RSA con SHA1. La primera parte identifica al criptosistema de clave pública utilizado mientras que la segunda parte identifica al algoritmo de hash utilizado para proteger la integridad del certificado.

- **Emisor**: Este campo contiene el nombre de la entidad que expidió el certificado digital. Este nombre es proporcionado de acuerdo al estándar X.500.

- **Validez**: El campo de validez indica la fecha a partir de la cuál el certificado es válido y la fecha en la cuál el certificado expira.

- **Sujeto**: El campo de sujeto contiene el nombre que identifica al propietario de la clave privada que corresponde a la clave pública contenida en el certificado. El sujeto puede ser cualquier entidad (usuario final, dispositivos de hardware, compañías, etc.).

- **Información de clave pública**: Este campo contiene la clave pública del sujeto, parámetros opcionales y el identificador del algoritmo. La clave pública contenida en este campo es utilizada para verificar las firmas digitales del sujeto.

- **Identificador único del emisor y del sujeto**: Estos campos contienen identificadores y aparecen en las versiones 2 o 3. Los identificadores del sujeto y del emisor son utilizados para la reutilización del nombre del emisor y el nombre del sujeto. En caso de existir dos entidades emisoras o dos sujetos con el mismo nombre, se puede utilizar este campo para desambiguar. Sin embargo, se ha probado que este mecanismo no es una solución satisfactoria. Actualmente el RFC 3280 no recomienda el uso de estos campos.

- **Extensiones**: Este es un campo opcional y aparece únicamente en los certificados X.509 v3. Si el campo está presente, el certificado contiene una o más extensiones de certificado; cada extensión incluye un identificador de extensión, una bandera que indica si la extensión es crítica o no, y el valor de la extensión. Comúnmente, las extensiones de los certificados han sido definidas por ISO y ANSI y la razón de su existencia es proporcionar mayor flexibilidad al certificado digital. Cualquier organización puede definir una extensión privada para cumplir con sus requerimientos específicos. Esta flexibilidad crea un problema: un certificado digital creado bajo el estándar X.509 v3 puede no ser totalmente legible por las implementaciones que soportan certificados X.509 v3. Cuando una extensión de certificado no es conocida por la aplicación que lo recibe, aparece la incompatibilidad. Esta es la razón de la existencia de la bandera indicando si una extensión es crítica o no lo es. Si es la extensión es marcada como no-crítica la aplicación lo único que hace es ignorar la extensión; por otra parte, si la extensión es marcada como crítica el resultado es que el certificado no puede ser utilizado debido a que se desconoce la funcionalidad de la extensión.

3.5. Extensiones X.509

La versión 2 de X.509 no se adapta a todos los requisitos que solicitan las aplicaciones actuales:

- El campo de identificación de sujeto y emisor es demasiado corto y no se adapta a algunas aplicaciones que se identifican con URL o e-mail.

- Es necesario agregar información de políticas de seguridad para poder ser utilizado por aplicaciones como IPSec.

- Es necesario acotar el daño producido por una CA defectuosa o maliciosa.

- Es necesario distinguir entre claves generadas por un mismo usuario en instantes de tiempo distintos.

En la versión 3[12] se propone introducir estas nuevas capacidades en forma de extensiones opcionales en lugar de campos fijos:

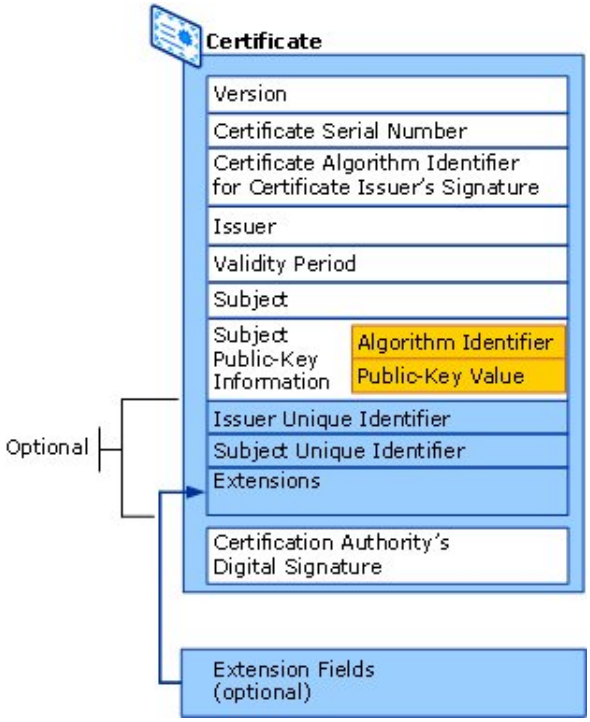

Figura 3.2: Formato de Certificado X.509 v3

Extensiones de información de políticas y claves, extensiones de atributos de emisor y sujeto y extensiones de restricciones del certificado. A continuación se listan las extensiones junto a una breve explicación de su utilidad.

- **Key Usage**: Esta extensión indica la finalidad de la clave pública certificada. Sus valores se suelen definir a través de plantillas de certificado.
- **Subject Alternative Name**: Esta extensión contiene uno o varios nombres alternativos, con diferentes formatos de nombre, para la entidad que la CA ha enlazado a la clave pública certificada. Las plantillas de certificado suelen definir cuál de estos formatos puede utilizarse en certificados que emplean esa plantilla.
- **Subject Key Identifier**: Esta extensión identifica qué clave pública se certifica mediante un certificado determinado. Su utilización principal es distinguir las claves cuando se certifican o se han certificado varias claves para la misma entidad.
- **Authority Key Identifier**: Esta extensión identifica qué clave pública fue utilizada por el emisor de un certificado al firmarla. Su utilización principal es distinguir las claves cuando se certifican varias claves para el mismo emisor. Su valor lo establece siempre la CA.
- **Private Key Usage Period**: Esta extensión restringe el uso de la clave privada que corresponde a una parte del período de validez del certificado. El perfil de certificado PKIX actual, RFC 2459, especifica que su uso ya no se recomienda.
- **Certificate Policies**: Esta extensión contiene una serie de indicadores de políticas. Un indicador de políticas puede constar solamente de un identificador de

objetos cuyo significado debe publicarse. De forma alternativa, puede constar de un identificador de objetos junto con una sentencia de la política prevista. La sentencia de política puede ser proporcionada por el URL desde el que puede recuperarse, o por una sentencia de texto breve incluida en el certificado.

- **Issuer Alternative Name**: Esta extensión contiene uno o varios nombres alternativos, con diferentes formatos de nombre, para el emisor del certificado. Su valor lo establece siempre la CA.
- **Subject Directory Attributes**: Esta extensión contiene una serie de atributos de directorio adicionales que pertenecen al sujeto y que no forman parte del nombre distintivo. Debe ser de tipo no crítico.
- **Extended Key Usage**: Esta extensión contiene una serie de identificadores de objetos que indican la finalidad de la clave pública certificada. Sus valores se suelen definir a través de plantillas de certificado. Los valores para esta extensión los puede definir una comunidad de usuarios o pueden derivarse de RFC 2459.
- **Basic Constraints**: Esta extensión es útil solamente para un certificado CA. Siempre está ausente o vacía para cualquier otro certificado, y RFC 2459 recomienda que esté ausente antes que vacía. Además de indicar que un certificado es un certificado CA, puede contener una ruta de acceso de longitud de certificación máxima, que especifica cuántos niveles más de CA pueden ser certificados por ésta. Esta extensión debe ser de tipo crítico.
- **Name Constraints**: Esta extensión se utiliza solamente en un certificado CA. Especifica un espacio de nombres dentro del cual deben estar ubicados todos los nombres de sujeto y los nombres de sujeto alternativos en los certificados que la CA o las CA certificadas por esa CA emiten junto con este certificado. El propósito de esta extensión es restringir los nombres que pueden utilizar los por certificados en la ruta de acceso de este certificado. Las restricciones se definen en términos de subárboles de nombres permitidos o excluidos. Un nombre que concuerde con una restricción en la lista de subárboles excluidos no es válido, con independencia de la información que aparezca en la lista de subárboles permitidos. Esta extensión debe ser de tipo crítico.
- **Policy Mappings**: Esta extensión se utiliza solamente en certificados CA.
- **Policy Constraints**: Esta extensión, que se utiliza solamente en un certificado CA, puede utilizarse para dos propósitos. Puede prohibir la correlación de políticas en los certificados de la ruta de acceso de certificación, o puede requerir políticas específicas para tales certificados.
- **CRL Distribution Points**: Esta extensión indica dónde puede hallarse una CRL parcial, que contiene información de revocación acerca de este certificado.
- **Authority Information Access**: Esta extensión indica dónde y cómo puede accederse a cierta información acerca del emisor del certificado en el que aparece la extensión.

3.6. Usos de los certificados

Según la RFC 2459, el formato de los certificados X.509 v3 contiene dos campos donde se puede indicar cuál va a ser la utilidad del certificado.

La extensión **Key Usage** define el propósito (por ejemplo: cifrado, firma) de la clave contenida en el certificado. Debe encontrarse presente y puede marcarse como crítica. Esta extensión debería ser empleada cuando se quiere restringir una o más operaciones que podrían ser realizadas usando las claves asociadas al certificado. Por ejemplo cuando una clave RSA debe ser solamente utilizada para firmar, deben marcarse los bits correspondientes a digitalSignature y nonRepudiation.

Los propósitos de la clave pueden ser los siguientes: firma digital, no repudio, cifrado de clave, cifrado de datos, acuerdo de clave, verificación de firma de CA en certificado, verificación de firma de CA en CRLs.

La extensión **Extended Key Usage** indica uno o más propósitos para los cuales el certificado puede ser utilizado, sumándose a los fines básicos que se indican en el campo explicado anteriormente. Los propósitos pueden ser definidos por una organización ante una necesidad y los identificadores de los objetos deben ser asignados de acuerdo a la normativa de la IANA o la ITU. Esta extensión puede ser crítica o no crítica. Si la extensión es marcada como crítica, el certificado debe ser utilizado solamente para uno de los fines indicados. La RFC 2459, define una serie de perfiles de uso de claves entre los cuales se encuentra uno que va a ser de gran utilidad en la implementación de este trabajo: **id-kp-timeStamping**.

3.7. PKCS

Otro estándar relativo con la infraestructura de PKI importante es PKCS (Public-Key Cryptography Standards)[13]. PKCS se refiere a un grupo de estándares criptográficos de clave pública diseñados y publicados por RSA Laboratories. PKCS provee interoperabilidad básica a aplicaciones que utilizan criptografía de clave pública. PKCS define los formatos de bajo nivel para el intercambio seguro de datos arbitrarios, tales como datos cifrados o firmados. Como lo indica el sitio web de RSA Laboratories, "Los estándares de criptografía de clave pública son especificaciones producidas por RSA Laboratories en cooperación con desarrolladores de sistemas seguros de todo el mundo, con el propósito de acelerar la implementación de la criptografía de clave pública".

3.8. Aplicaciones de la firma digital

- **Cifrado de comunicaciones**: La aplicación más extendida de la firma digital es la comunicación segura entre servidores web cuando se utiliza https como protocolo de transporte. En este caso, el navegador solicita un certificado digital al sitio, y este caso el sitio debe presentar un certificado firmado por una de las entidades reconocidas por el navegador. Entonces, se establece una comunicación asimétrica para acordar una clave secreta y un algoritmo de cifrado y a partir de ese momento, se establece una comunicación cifrada simétrica.

- **Firma de mensajes y documentos**: Si un usuario posee un certificado digital

instalado en un programa de correo, puede enviar mensajes de correo electrónico firmados digitalmente. Esto permite a los destinatarios verificar la autenticidad del correo y por lo tanto se obtiene la garantía de que nadie ha suplantado la identidad del emisor. Dado que suplantar la identidad del emisor del correo electrónico resulta sencillo por la baja seguridad de los protocolos utilizados, esta aplicación resulta muy interesante.

- **Identificación ante un sistema o autenticación de usuarios**: La forma más común de identificarse ante un sistema informático es mediante la utilización de un nombre de usuario y una contraseña. En situaciones donde los niveles de seguridad son determinantes, la utilización de certificados digitales para autenticarse en un sitio es un mecanismo que se está comenzando a utilizar con mayor asiduez. En el momento de ingresar al sitio en cuestión, el navegador o la interfaz del sistema le solicita al usuario que le presente su certificado digital.

Capítulo 4

Tiempo

El "Tiempo" es la magnitud física con la que se mide la duración o separación de acontecimientos sujetos a cambio, de los sistemas sujetos a observación. Es la magnitud que permite ordenar los sucesos en secuencias, estableciendo un pasado, un presente y un futuro.

Si analizamos la definición física de tiempo, dados dos eventos puntuales E1 y E2, que ocurren respectivamente en instantes de tiempo t1 y t2, y en puntos del espacio diferentes P1 y P2, todas las teorías físicas admiten que éstos pueden cumplir una y sólo una de las siguientes tres condiciones:

1. Es posible para un observador estar presente en el evento E1 y luego estar en el evento E2, y en ese caso se afirma que E1 es un evento anterior a E2. Además, si eso sucede, ese observador no podrá verificar la condición 2.

2. Es posible para un observador estar presente en el evento E2 y luego estar en el evento E1, y en ese caso se afirma que E1 es un evento posterior a E2. Además si eso sucede, ese observador no podrá verificar la condición 1.

3. Es imposible, para un observador puntual, estar presente simultáneamente en los eventos E1 y E2.

Dado un evento cualquiera, el conjunto de eventos puede dividirse según esas tres categorías anteriores. Es decir, todas las teorías físicas permiten, fijado un evento, clasificar a los eventos en: (1) pasado, (2) futuro y (3) resto de eventos (ni pasados ni futuros). La clasificación de un tiempo presente es debatible por la poca durabilidad de este intervalo que no se puede medir como un estado actual sino como un dato que se obtiene en una continua sucesión de eventos.

La cronología (histórica, geológica, etc.) permite datar los momentos en los que ocurren determinados hechos (lapsos relativamente breves) o procesos (lapsos de duración mayor). En una línea de tiempo se puede representar gráficamente los momentos históricos en puntos y los procesos en segmentos.

Las formas e instrumentos para medir el tiempo son de uso muy antiguo, y todas ellas se basan en la medición del movimiento, del cambio material de un objeto a través del tiempo, que es lo que puede medirse. En un principio, se comenzaron a medir los movimientos de los astros, especialmente el movimiento aparente del Sol, dando lugar al tiempo solar aparente. El desarrollo de la astronomía hizo que, de manera paulatina,

se fueran creando diversos instrumentos, tales como los relojes de sol, los relojes de arena y los cronómetros. Posteriormente, la determinación de la medida del tiempo se fue perfeccionando hasta llegar al reloj atómico.

En las secciones posteriores, se explicará el concepto de escala de tiempo, se presentarán diversas escalas de tiempo que se consideran importantes para entender el concepto de tiempo aplicado al sistema de sellado de tiempo digital, y por último se presentará NTP, un protocolo que permite la sincronización de relojes en sistemas informáticos.

4.1. Escalas de tiempo

A grandes rasgos, en la actualidad podemos dividir las escalas de tiempo[14] en dos familias: las asociadas al movimiento de cuerpos celestes (incluyendo la rotación de la Tierra), y las basadas en la oscilación de átomos. En las siguientes secciones se describirán algunas de las escalas más utilizadas de las basadas en la oscilación de los átomos, que van a hacer de utilidad para enriquecer el desarrollo de este prototipo.

4.1.1. Tiempo atómico internacional

El Tiempo Atómico Internacional es un estándar atómico de alta precisión para medir el tiempo propio de un cuerpo geoide con un reloj atómico. Es una escala de tiempo continuo y constante. Su unidad es el segundo atómico definido como la unidad vigente del Sistema Internacional, y su valor es el correspondiente a 9192631770 períodos de la radiación correspondiente a la transición entre dos niveles hiperfinos en el átomo de cesio 133. Esto significa, que por primera vez la unidad de tiempo no está ligada a un fenómeno astronómico.

La rotación de la Tierra no es uniforme. Se ha comprobado que nuestro planeta gira cada vez más lentamente de forma que el día actual es aproximadamente 16 milisegundos más largo que hace 1000 años. Además, la nutación de los polos y las fluctuaciones en la inclinación de la Tierra introducen perturbaciones de algunos milisegundos al año. Todos estos efectos contribuyen a que la Tierra sea un reloj irregular e inexacto, y ha dado lugar al desarrollo de 4 escalas de tiempo que se denominan de forma genérica Tiempo Universal.

4.1.2. GMT

El Tiempo Medio de Greenwich es el tiempo solar medio en el Observatorio Real de Greenwich[15], cerca de Londres, que por convención se encuentra a cero grados de longitud. Durante mucho tiempo, los relojes más precisos que existían eran el movimiento de la Tierra alrededor de su eje y alrededor del Sol. A partir de ellos, se determinaba todo lo relacionado con el tiempo. La vuelta de la Tierra alrededor del Sol determinaba la duración de un año y el tiempo que tardaba la tierra en girar sobre sí misma determinaba la duración de un día, el cual se dividía en 24 horas, la hora se dividía en 60 minutos y cada minuto consistía de 60 segundos. En el año 1900 se determinó que un segundo equivalía a 1/86.400 de un día solar medio.

En un principio, se consideró que esta manera de determinar y medir el tiempo era la correcta, pero varios estudios determinaron que diversos factores aleatorios generaban diferencias en la duración de los procesos de translación y rotación.

A medida que se comenzó a estudiar los átomos, se descubrió un reloj mucho más preciso, que se basa en los cambios de estados en los átomos. En el año 1950, en Estados Unidos, se construyó el primer reloj atómico, construido en base a átomos de cesio. Su precisión era tan alta que en 1967 los organismos de normas internacionales cambiaron la definición de segundo basada en el movimiento de la Tierra por una definición basada en el átomo de cesio: el segundo, unidad de tiempo del Sistema Internacional de Unidades, es la duración de 9.192.631.770 períodos de la radiación asociada a la transición hiperfina del estado base del átomo de cesio 133, con la siguiente observación: el estado base se define con campo magnético cero.

4.1.3. UTC

El Tiempo Universal Coordinado (UTC)[16] es la escala de tiempo que determina la hora exacta para los husos horarios mundiales, siendo el sucesor de GMT. A diferencia de GMT, esta escala utiliza relojes atómicos. La Oficina Internacional de Pesas y Medidas (BIPM)[17] es el organismo que a nivel internacional genera y mantiene la escala de tiempo UTC. Las realizaciones que los diferentes países mantienen a partir de UTC por medio de sus laboratorios primarios de metrología, las cuales son denotadas por UTC(k), donde k indica el acrónimo del laboratorio nacional, son escalas de tiempo de suma importancia ya que éstas son las que se utilizan para las aplicaciones directas en los procesos productivos. En la República Argentina, contribuyen a la formación de UTC el Observatorio Naval de Buenos Aires[18] y el Instituto Geográfico Militar[19].

Dado que el giro de la Tierra es menos uniforme que el comportamiento de los relojes atómicos, hay una cierta discrepancia entre el tiempo solar medio, base del GMT, y el UTC. Para que haya sincronía entre los dos tiempos, lo que se hace es controlar con extrema precisión el giro de la Tierra. Se admite que UTC y GMT son correctos si no difieren en más de 0,9 segundos. Si difieren en más de esa cantidad, se añade o se quita un segundo a los relojes atómicos.

4.2. GPS

El Navstar Global Positioning System (GPS)[20] fue desarrollado en los años setenta por las fuerzas militares de Estados Unidos con el objetivo de brindar posicionamiento e información del tiempo en todo el mundo. El sistema consiste de 28 satélites que se encuentran orbitando alrededor del globo en seis grupos con cuatros satélites en cada uno, aproximadamente a 20000 km de distancia. Cada satélite cuenta con 4 relojes atómicos, para lograr mayor precisión.

Para mantener la precisión, la mayoría de los satélites requieren actualizaciones diarias de sus datos.

Al momento de determinar la posición de un objeto, es necesario localizar tres satélites, y teniendo en cuenta la información obtenida de los mismos, se utiliza un mecanismo de triangulación para calcular la posición en que se encuentra el mismo. Si

se utilizan simultáneamente más satélites, la precisión de la posición obtenida se mejora notablemente.

Cada uno de los satélites que conforman el sistema de navegación, dispone a bordo de dos relojes atómicos de cesio que en todo momento marcan el tiempo universal y emiten una señal horaria marcando el comienzo de cada segundo de tiempo universal.

4.3. NTP

NTP[21] es un protocolo de Internet para sincronizar los relojes de los sistemas informáticos a través del enrutamiento de paquetes en redes con latencia variable. NTP utiliza UDP como protocolo de capa de transporte, utilizando el puerto 123. Está diseñado para resistir los efectos de la latencia variable.

NTP utiliza el algoritmo de Marzullo con la escala de tiempo UTC, incluyendo soporte para características como segundos escalares. NTPv4 puede mantenerse sincronizado con un diferencia máxima de 10 milisegundos(1/100 segundos) a través de Internet, y puede llegar hasta 200 microsegundos (1/5000 segundos) o más en redes de área local sobre condiciones ideales.

Este protocolo es uno de los protocolos más viejos que siguen en uso (desde antes de 1985). NTP fue diseñado por Dave Mills, quien lo sigue manteniendo junto a un grupo de voluntarios.

NTP utiliza un sistema de jerarquía de estratos de reloj, en donde los sistemas de estrato 1 están sincronizados con un reloj externo tal como un reloj GPS ó algún reloj de radio. Los sistemas de estrato 2 de NTP derivan su tiempo de uno ó más de los sistemas de estrato 1, y así consecutivamente (cabe mencionar que ésto es diferente de los estrato de reloj utilizados en los sistemas de telecomunicaciones).

Las estampas de tiempo utilizadas por NTP consisten en un segundo de 32-bit y una parte fraccional de 32-bit, dando con esto una escala de 232 segundos (136 años), con una resolución teórica de 2 32 segundos (0.233 nanosegundos). Aunque las escalas de tiempo NTP se redondean cada 232 segundos, las implementaciones deberían desambiguar el tiempo NTP utilizando el tiempo aproximado de otras fuentes. Esto no es un problema en la utilización general ya que esto solamente requiere un tiempo cercano a unas cuantas décadas.

Los detalles operacionales de NTP se encuentran ilustrados en el RFC 778[22], RFC 891[23], RFC 956[24], RFC 958[21] y RFC 1305[25]. (NTP no debe ser confundido con daytime (RFC 867)[26] ó los protocolos de tiempo (RFC 868)[27]). La versión actual de NTP es la versión 4; hasta el 2005, sólo las versiones superiores a la versión 3 han sido documentadas en los RFCs. El grupo de trabajo de NTP IETF ha sido formado para estandarizar el trabajo de la comunidad de NTP desde RFC 1305.

NTP utiliza un sistema jerárquico basado en capas de fuentes de reloj. Cada nivel de esta jerarquía se denomina estrato y se les asigna un número de capa comenzando por el cero en la capa superior. El nivel del estrato define la distancia desde el reloj de referencia y existe para prevenir las dependencias cíclicas en la jerarquía. Es importante señalar que el estrato no es una indicación de la calidad o fiabilidad, es muy común encontrar fuentes de tiempo "estrato 3" que son de mayor calidad que otra fuente de tiempo de "estrato".

- **Stratum 0**: Son dispositivos como relojes atómicos, GPS o de radio. Generalmen-

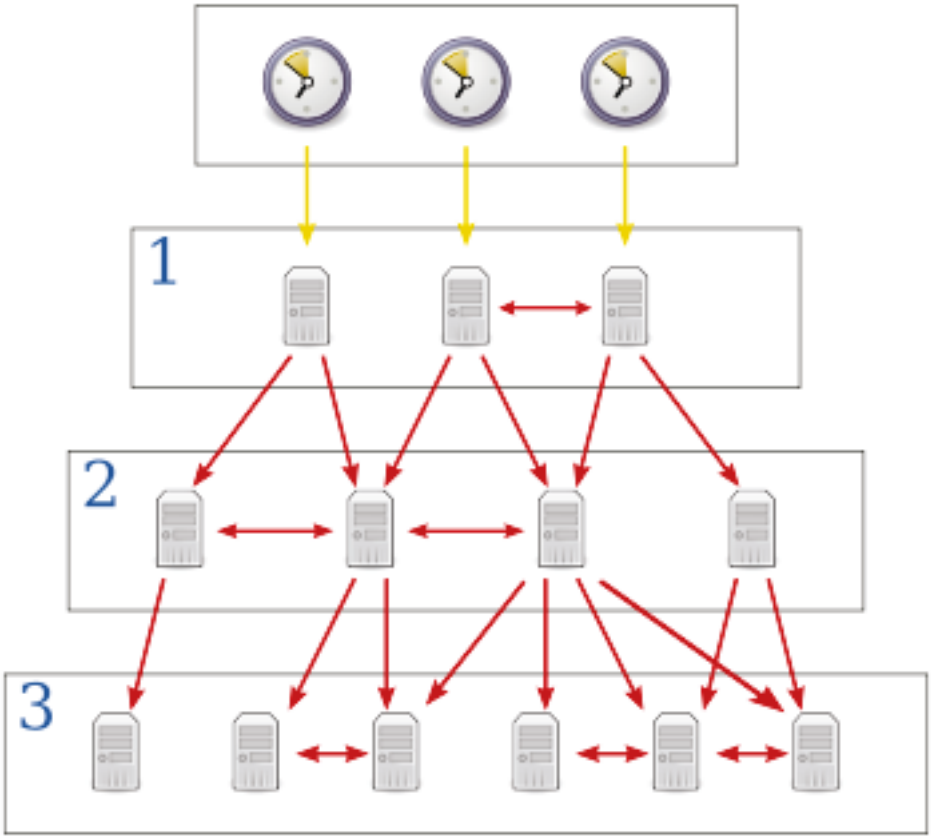

Figura 4.1: Estructura de NTP

te este tipo de relojes no está conectado a la red, sino que se encuentra conectado a una computadora mediante una interfaz.

- **Stratum 1**: Son computadoras conectadas a dispositivos de Stratum 0. Normalmente, estos actúan como servidores para las solicitudes de tiempo desde servidores Stratum 2 mediante el protocolo NTP. Estas computadoras se conocen generalmente como servidores de tiempo. Existen gran cantidad de servidores de Stratum 1 diseminados alrededor del mundo.

- **Stratum 2**: Son computadoras que envían peticiones NTP a servidores Stratum 1. Normalmente una computadora Stratum 1 puede hacer referencia a varios servidores de nivel superior y utilizar el algoritmo NTP para obtener la mejor muestra, eliminando los servidores que considera que se encuentran mal calibrados. Los servidores de Stratum 2 pueden dialogar con otros de su mismo nivel para proveer más estabilidad y robustez sobre el tiempo para todos los pares. Generalmente, estos equipos trabajan como servidores para solicitudes desde el Stratum 3.

- **Stratum 3**: Estos equipos emplean exactamente las mismas funciones NTP de interconexión y los datos de muestreo del Stratum 2, y pueden ellos mismos actuar como servidores de los estratos más bajos, posiblemente hasta 16 niveles. NTP (dependiendo de la versión del protocolo NTP en uso) soporta hasta 256 capas.

Capítulo 5

Descripción del Sellado de Tiempo

El servicio de Time Stamping se sustenta en los mecanismos de firma digital y generalmente es un servicio adicional que prestan las autoridades de certificación. A grandes rasgos, existe una tercera parte de confianza, que es aceptada tanto por el emisor como por el receptor, que es la que da fe de la fecha y hora de una transacción. Es decir, añade el dato "tiempo" a la transacción o al documento, por el cual las partes aceptan la validez temporal que se asocia a ese dato determinado.

A lo largo de este capítulo se presentarán diversos tópicos que ayudarán a comprender la importancia de este servicio y cuáles son los beneficios de contar con una Autoridad de Sellado de Tiempo dentro de una Autoridad de Firma Digital ya implementada.

Al inicio de este capítulo, se especifican y analizan los distintos estándares relacionados con el sellado de tiempo. Luego se detallan las fases involucradas en el sellado de tiempo y los roles que se pueden distinguir en el marco del proceso. Una vez explicados los conceptos teóricos asociados que se consideran esenciales para comprender el funcionamiento del servicio, se explicarán conceptos más avanzados como la verificación de los sellos a largo plazo, el desarrollo de las políticas y procedimientos asociados al servicio y los aspectos de seguridad que la normativa existente insta a cumplimentar. El capítulo continuará presentando una problemática importante relacionada con la emisión de sellos de tiempo y el análisis de las necesidades actuales a nivel global que conllevan a desarrollar este tipo de servicios. Para finalizar se presentará un cuadro comparativo de los diversos estándares involucrados en la definición del servicio de sellado de tiempo.

5.1. Estándares involucrados

5.1.1. RFC 3161

La RFC 3161[28] describe el formato de una petición enviada a una autoridad de sellado de tiempo y el formato de la respuesta obtenida. Además establece gran cantidad de requisitos de seguridad relevantes para la operación de la autoridad, en lo que respecta a la generación de las solicitudes para obtener las respuestas.

Esta RFC fue actualizada por la RFC 5816[29] en marzo de 2010, con pequeñas modificaciones que actualizan la RFC al estado del arte actual de la criptografía.

5.1.2. ISO/IEC 18014

El estándar ISO/IEC 18014 es un estándar internacional que especifica diversas técnicas de sellado de tiempo. El mismo consiste de tres partes, las cuales incluyen nociones generales, modelos para el servicio de sellado de tiempo y estructuras de datos y protocolos.

ISO/IEC 18014-1:2008

La parte inicial de la ISO/IEC 18014-1[30] describe el marco de trabajo y define las nociones básicas, las estructuras de datos y los protocolos que son utilizados para las diversas técnicas de sellado de tiempo.

- Identifica el objetivo de una autoridad de sellado de tiempo.
- Describe un modelo general en el cuál se basan los servicios de sellado de tiempo.
- Describe el proceso de generación y verificación de los sellos de tiempo.
- Describe las estructuras de datos involucradas en los token de sellado de tiempo.
- Especifica los protocolos utilizados entre las entidades involucradas.

ISO/IEC 18014-2:2009

La segunda parte de la ISO/IEC 18014[31] presenta un marco general para el aprovisionamiento de un servicio de sellado de tiempo.

Los servicios de sellado de tiempo pueden generar, renovar y verificar los token de sellado de tiempo.

Los tokens son asociaciones entre un dato e instantes determinados de tiempo, y son creados de forma tal que permiten proporcionar pruebas de que el dato existió en una determinada fecha y hora. Además dicha evidencia puede ser utilizada en los servicios de no repudio.

Además, esta sección especifica mecanismos que generan sellos de tiempo independientes, lo que permite en el momento de verificar un sello de tiempo no tener la necesidad de acceder a otros sellos de tiempo vinculados.

ISO/IEC 18014-3:2009

La tercera parte de la norma creada por la ISO[32] comprende las siguientes características:

- Describe un modelo general para producir tokens vinculados.
- Describe los componentes básicos utilizados para construir un servicio de sellado de tiempo y producir tokens vinculados.
- Define las estructuras de datos utilizadas para interactuar con un servicio de sellado de tiempo en la producción de tokens vinculados.
- Describe instancias específicas de los servicios de sellado de tiempo y define los protocolos que serán utilizados en este modelo de servicio con el objetivo de extender el modelo existente.

5.1.3. ANSI X9.95

Este estándar desarrollado por la ANSI[33] especifica los requisitos mínimos de seguridad para el uso efectivo de los sellos de tiempo en un entorno de servicios financieros. Dentro del alcance de este estándar, los siguiente tópicos son tratados:

- Requerimientos para la manipulación segura de los sellos de tiempo a lo largo de su ciclo de vida, comprendiendo los procesos de generación, transmisión y almacenamiento, validación y renovación.
- Requerimientos para el manejo seguro de una autoridad de sellado de tiempo.
- Requerimientos para la autoridad de sellado de tiempo para asegurar que terceras partes puedan auditar y validar los controles sobre los procesos involucrados en el sellado de tiempo.
- Técnicas utilizadas y recomendadas para la codificación, encapsulamiento, transmisión y almacenamiento de los datos involucrados.
- Uso de la tecnología de sellado de tiempo.

5.1.4. ETSI TS 101 861 "Time Stamping Profile"

La especificación de la ETSI TS 101 861[34] está basada en el estándar de la RFC 3161. Define lo que debe soportar un cliente que va a hacer uso del servicio de sellado de tiempo y qué debe soportar un servidor de sellado de tiempo.

5.1.5. ETSI TS 102 023 ó RFC 3628

Este documento que fue creado por la ETSI y luego adoptado por la IETF como la RFC 3628[35] se puede definir como una guía de asistencia en la redacción de la política de sellado de tiempo para autoridades de sellado de tiempo que generan tokens, utilizando una infraestructura de clave pública con una exactitud mínima de un segundo. Una autoridad puede definir su propia política, basándose en la política definida en este documento. La misma podrá incorporar o limitar los requisitos definidos en esta RFC.

5.2. Fases

En el proceso de sellado de tiempo se pueden distinguir dos fases:

5.2.1. Emisión del sello de tiempo

En primer término, el solicitante genera un resumen de la información que quiere sellar. Este hash es enviado a la autoridad de sellado de tiempo, la cual anexa el sello de tiempo tiempo al hash y vuelve a calcular el resumen considerando ahora el nuevo dato generado. Este hash es firmado digitalmente con la clave privada de la TSA. Por último el hash firmado junto con el sello de tiempo son enviados al solicitante del sellado de tiempo.

Según la RFC 3161, el solicitante deberá enviar una petición TimeStamp Request que deberá incluir estos parámetros en forma obligatoria:

- Hash del documento a sellar.
- Nombre del algoritmo de hash a utilizar.
- OID de política bajo la cual se proporcionará el sello.

Por su parte, durante el proceso de sellado, la autoridad de sellado realiza diferentes acciones antes de emitir el sello de tiempo. En primer término, analiza la petición, verificando la correcta estructuración del objeto TimeStamp Request y el origen de la misma. Durante este chequeo se comprueba que el tanto el algoritmo de hash seleccionado como la política de sellado aplicable sean aceptables, según lo reglamentado en la política de sellado de la Autoridad de sellado de tiempo.

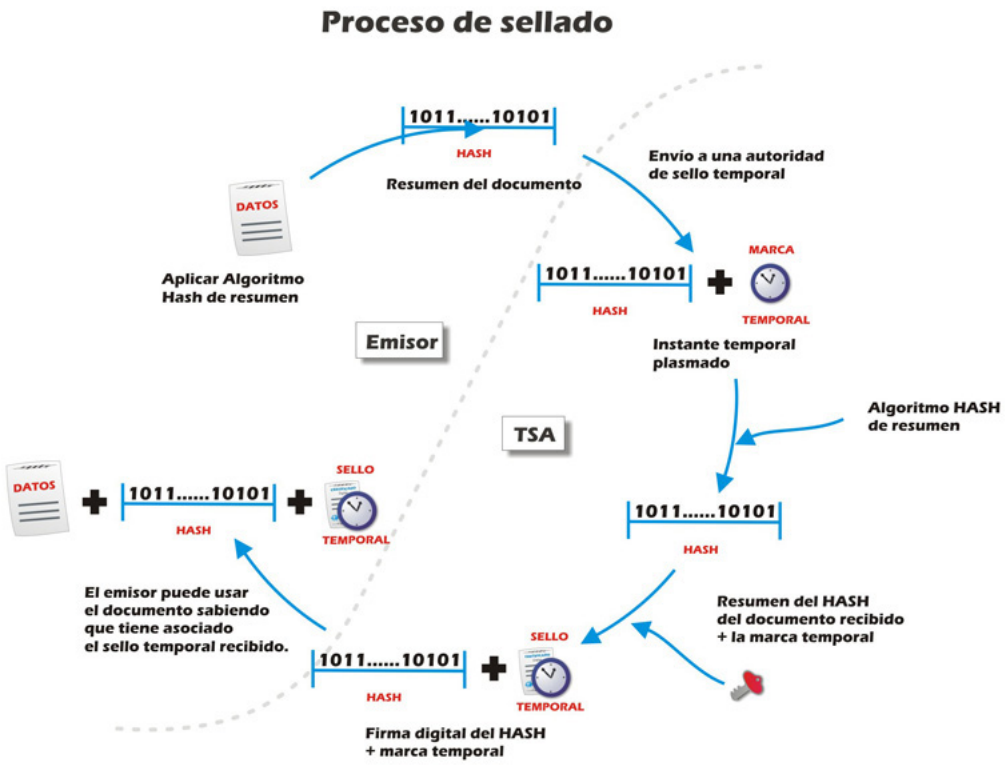

Figura 5.1: Obtención de sello de tiempo. Fuente: xolido.com

5.2.2. Verificación del sello de tiempo

Cualquier entidad que confíe en el emisor del sello de tiempo puede verificar que el documento no fue creado después de la fecha que indica el sello. Para probar esto, se calcula el hash de la información original, se concatena a este hash el sello de tiempo recibido y se vuelve a calcular una nueva función de hash.

En este punto, resta validar la firma digital de la TSA. Se debe verificar que el hash recibido fue firmado con la clave privada. Para ello, se aplica la clave pública de la TSA

a dicho dato, y se comparan ambos hash. Esta comprobación permite probar que el sello de tiempo y el mensaje no fueron alterados y que efectivamente fue emitido por la autoridad de sellado.

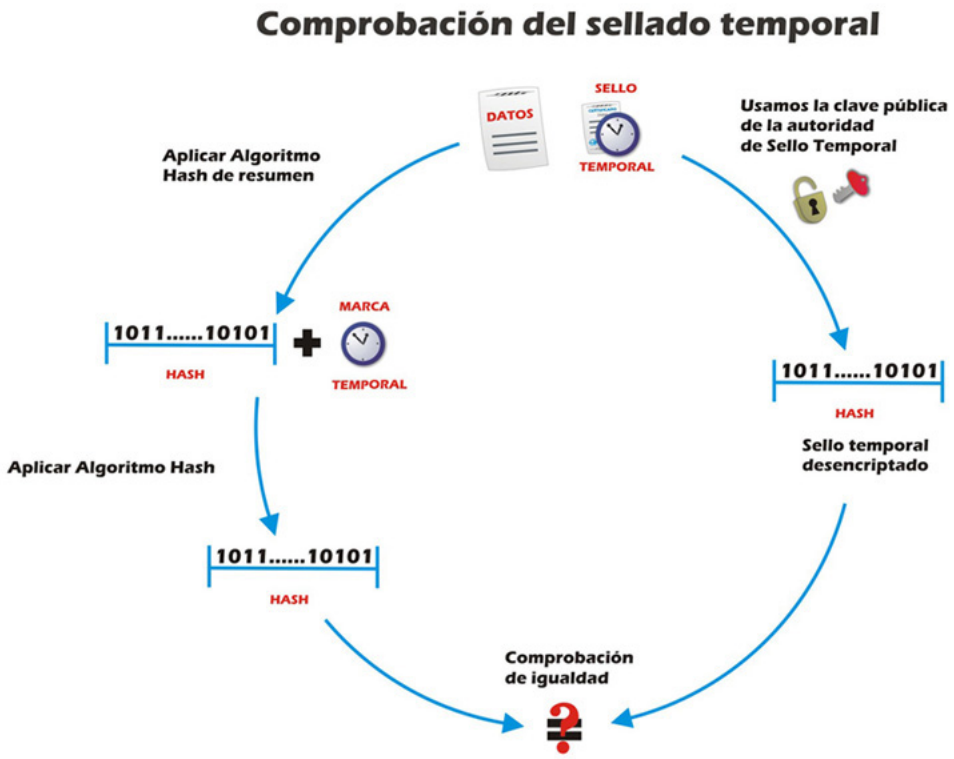

Figura 5.2: Verificación de un sello de tiempo. Fuente: xolido.com

5.3. Roles

5.3.1. TSA

La autoridad de sellado de tiempo, TSA (Time Stamping Authority, por sus siglas en inglés) es el proveedor del servicio. Su finalidad es la de comprobar la existencia de los datos a sellar y generar el sello de tiempo que irá unido a esos datos. De esta forma, la TSA asegura que esos datos existían en un determinado instante de tiempo y garantiza que el parámetro de tiempo de ese sello es correcto.
Según lo establecido por la IETF, una TSA debe:

- Utilizar una fuente fiable de tiempo.
- Incluir un valor de tiempo fiable para cada sello emitido.
- Incluir un valor entero único para cada nuevo sello generado, conocido como nonce, por la abreviatura en inglés de *number used once*.
- Producir un nuevo sello de tiempo cada vez que se reciba una petición válida de un solicitante.

- Incluir en cada sello un identificador que indica la política de sellado de tiempo utilizada para la creación del mismo.
- Verificar el algoritmo de resumen, chequeando que se encuentre dentro de los aceptados para la política de seguridad que se está aplicando y que la longitud del dato enviado se corresponda con lo esperado según el algoritmo utilizado.
- No examinar el dato que va a ser sellado de ninguna forma, excepto las mencionadas anteriormente.
- No incorporar ninguna identificación del solicitante en el sello emitido.
- Firmar cada sello utilizando una clave generada exclusivamente con ese fin, indicando esta propiedad en el certificado digital correspondiente.
- Anexar información al sello solicitado por el cliente utilizando los campos de extensión, teniendo en cuenta las extensiones que la TSA en cuestión acepte.

5.3.2. Solicitante

Es la entidad que posee documentos, información o, en general, cualquier tipo de datos electrónicos a los que quiere incluir un sello de tiempo que garantice que fueron creados previo a la solicitud del sello. La solicitud contiene el hash del dato al cual se le quiere estampar el tiempo.

5.3.3. Fuente de tiempo

La TSE, por sus siglas en inglés Time Stamp Source Entity, es cualquier Instituto Nacional de Mediciones, como por ejemplo el NIST o el USNO, que provee al sistema de los servicios de calibración del tiempo. El origen del tiempo para cualquier instituto de medición es la Autoridad Internacional del Tiempo (ITA). Actualmente, la Oficina Internacional de Pesas y Medidas es el coordinador mundial de la metrología y es el que se encarga de calibrar los relojes de cada instituto nacional de mediciones. Otro componente relacionado con el tiempo que únicamente es mencionado en la normativa ANSI X9.95 es el Reporte de Calibración del Tiempo (TCR). El mismo es utilizado entre la TSA y la TSE, proveyendo mecanismos de auditoría en los eventos de calibración del tiempo.

5.3.4. Verificador

Es la entidad que quiere comprobar que los datos sellados que ha recibido contienen un sello de tiempo válido. Incluso podría ser la misma entidad que utilizó el servicio de sellado de tiempo, para comprobar que el sello generado es válido y correcto.

Un verificador se encuentra obligado a comprobar:

- que el sello de tiempo fue correctamente firmado y que no se ha revocado el certificado correspondiente;
- que la política de sello de tiempo de la TSA no establece limitaciones en cuanto a la aplicabilidad de los sellos de tiempo incompatibles con el uso dado; y

- toda otra precaución surgida de contratos o convenios vinculados.

5.4. Políticas y procedimientos

Una política de sellado de tiempo está conformada por una serie de reglas que indican la aplicabilidad de un sello de tiempo a una comunidad en particular o clase de aplicaciones, que poseen requerimientos similares de seguridad y funcionalidad.

Una política de Sello de Tiempo se define independientemente de los detalles operativos específicos de la instalación, mientras que los procedimientos se desarrollan en función del ambiente de sistemas, de la organización y de las instalaciones.

En líneas generales, una política es un documento con menor nivel de especificidad que determina a lo que se adhiere, mientras que los procedimientos establecen con detalle la forma en que se llevan a a cabo las tareas.

Además, existen documentos adicionales, descriptos en el estándar de la ETSI entre los que se pueden encontrar la Política de Privacidad y el Acuerdo con Suscriptores.

5.5. Verificación de sellos de tiempo a largo plazo

Generalmente, un sello de tiempo emitido se vuelve inverificable una vez transcurrido el período de validez del certificado que ha firmado dicho dato. Esto se debe a que la autoridad de certificación que ha emitido el certificado utilizado para la firma, deja de garantizar que va a publicar los datos de revocación, los cuales incluyen las revocaciones causadas por claves comprometidas.

Sin embargo, la verificación de un sello de tiempo bajo las condiciones especificadas anteriormente, puede realizarse si se puede asegurar las siguientes condiciones:

- La clave privada de la TSA no ha sido comprometida hasta la fecha en que se realiza la verificación del sello.
- Los algoritmos de hashing utilizados no han presentado colisiones hasta el momento de la verificación.
- El algoritmo de firma y el tamaño de la clave de firma se encuentran fuera del alcance de los ataques criptográficos al momento de la verificación.

Si alguna de estas condiciones no se puede garantizar, entonces la validez puede ser mantenida mediante la aplicación de un resellado, que consiste en volver a aplicar un sello de tiempo, para proteger la integridad de los datos previamente sellados.

5.6. Aspectos de seguridad de la TSA

La RFC 3161 realiza diversas consideraciones relacionadas con la seguridad de una TSA:

1. Cuando una TSA no va a ser utilizada más, pero la clave privada de la misma no fue comprometida, debe procederse en forma inmediata a la revocación del certificado digital. En relación a esta cuestión, la RFC menciona cuestiones técnicas

que deben realizarse en relación al registro del evento en los repositorios de la Autoridad de Certificación involucrada.

2. Cuando la clave privada de la TSA sea comprometida, entonces el certificado debe revocarse inmediatamente. Cualquier sello firmado por la TSA utilizando la clave privada involucrada no debe considerarse válido. Por lo tanto, es imperativo que la clave privada de la TSA sea resguardada con un nivel adecuado de seguridad para minimizar los riesgos del compromiso de la misma.

3. La clave de la TSA debe ser de un longitud suficientemente larga para permitir un tiempo de vida extenso. Sin embargo, la clave va a tener un tiempo de validez finito, por lo tanto se indica que los sellos deben ser sellados nuevamente para poder mantener la confiablidad de los mismos.

4. Una aplicación cliente que utiliza un "nonce" y no un reloj local, debe considerar el tiempo de espera que está dispuesto a aceptar para recibir una respuesta, para evitar sufrir un ataque del estilo "man-in-the-middle". Por lo tanto, cualquier respuesta que lleva un tiempo mayor que el considerado adecuado, debe tratarse como sospechosa. El período de tiempo considerado varía según el método de transporte utilizado, entre otros factores relativos a la configuración del servicio.

5. Si varias entidades pueden obtener sellos de tiempo otorgados sobre el mismo objeto o dato, utilizando el mismo algoritmo, o bien, si una misma entidad puede tener varios sellos de tiempo sobre el mismo dato, el sello de tiempo generado tendrá el mismo valor de hash. Por lo tanto, un tercero con acceso a los sellos de tiempo puede deducir que estos sellos de tiempo se corresponden a la misma información o dato original.

6. Es posible que se reenvíe un mismo requerimiento de sello de tiempo más de una vez a una misma TSA, debido a problemas en la red. También puede pasar que un sello de tiempo se emita más de una vez. Para detectar estas situaciones, existen diversas técnicas, como el uso del "nonce" y el uso de un reloj local y una ventana de tiempo durante la cual se registran todos los hashes enviados con el fin de detectar cualquier posible repetición.

5.7. El problema con los sellos de tiempo

En los sistemas de sellado digital de tiempo, existen redes que proveen un sistema de tiempo de forma tal que el tiempo puede ser estampado con el año, mes, día, hora, minutos y segundos. Este tiempo es agregado al dato en cuestión y luego el documento entero es firmado digitalmente. Seguramente, el sello de tiempo indica un secuencia particular de eventos y permite determinar cuando un documento fue creado. Aunque es importante observar que los sellos de tiempo generados no son independientes de la generación de datos o de los procesos de generación de la firma digital.

El reloj del sistema, a través de una red de área local o Internet, puede ser sincronizado con un instituto nacional de medición cuyo reloj es calibrado con la autoridad del tiempo, que es el Bureau International des Poids et meisures. El problema se genera debido a que el administrador del sistema puede modificar el reloj del sistema seteando

el valor que él desee. Es posible generar sellos de tiempo que no se correspondan con el tiempo en el que dato fue firmado.

Por ejemplo, un administrador puede manipular el reloj del sistema por lo que un mismo conjunto de datos, firmados en instantes de tiempo diferentes, pueden tener el mismo sello de tiempo.

En este marco, aunque se cuente con una infraestructura de firma digital y terceras partes que confían en la misma, no se puede confiar en los datos firmados, debido a que no existen mecanismos que permitan distinguir entre las distintas versiones generadas y no hay métodos para poder armar la secuencia temporal de versiones correctamente. Esto sucede porque la firma digital no es sensible al tiempo; por lo tanto no es posible confiar en los sellos de tiempo solamente porque estén firmados digitalmente.

Para poder distinguir entre las diversas versiones y confiar en la firma digital en los sellos de tiempo, se requiere otra aproximación al problema. Es necesario un mecanismo para generar sellos de tiempo verificables que sean independientes de la firma digital, lo que se conoce como sellado de tiempo confiable.

5.8. Aplicaciones del sellado digital de tiempo

El sellado digital de tiempo puede ser utilizado en una gran cantidad de áreas de aplicación entre las que se destacan las siguientes:

- Comercio electrónico.
- Protección de la propiedad intelectual.
- Transacciones bursátiles.
- Protección de documentos electrónicos contra la falsificación y el cambio de fecha de creación.
- Envío de información en formato electrónico a organismos gubernamentales.
- Protección de archivos de logging en sistemas informáticos, con el fin de protegerlos contra la adulteración.
- Licitaciones o concursos públicos de ofertas a través de medios electrónicos.
- Firma de documentos y contratos.
- Cierre de libros financieros.
- Declaraciones de testamentos.

5.9. Necesidades del sellado de tiempo

5.9.1. Incrementar la confianza en el comercio electrónico

La precisión en el tiempo y los contenidos de las transacciones resulta de gran importancia en el comercio electrónico. Sin embargo, las transacciones actuales se realizan

usando fuentes de tiempo de los propios dispositivos de compradores o vendedores, por tanto el tiempo no es confiable y puede ser fácilmente manipulado y repudiado. Asimismo, los contenidos de los pedidos, facturas, u otros documentos implicados en transacciones on-line son susceptibles de ser alterados. Estos problemas diezman la confianza de los usuarios en el comercio electrónico y entorpecen su desarrollo.

Aplicando sellos de tiempo en los documentos manejados on-line se garantiza que las transacciones ocurren en un momento particular y sus contenidos no han sido alterados desde entonces. Integrando un servicio de sellado de tiempo, el fraude y el repudio no son factibles. De esta manera, los compradores y vendedores pueden operar en un entorno fiable y confiar en el comercio en Internet.

5.9.2. Para proteger la propiedad intelectual

Internet es un medio adecuado para compartir el trabajo creativo. Sin embargo, desafortunadamente, cualquier trabajo publicado en Internet es susceptible de ser plagiado. Además, los mecanismos para demostrar la autoría de un trabajo, en caso de llegar a una disputa, son prácticamente inexistentes.

El sellado de tiempo puede usarse para certificar la existencia de cualquier trabajo creativo, incluyendo texto, gráficos, audio o video, desde de un momento concreto.

Emitir sellos de tiempo electrónicos impide infringir los derechos de la propiedad intelectual. Si la creación se plagia, la persona con el sello de tiempo más antiguo tendrá una prueba fehaciente para reclamar la propiedad del copyright de esa creación.

5.9.3. Para soportar la infraestructura de PKI existente

En la existente Infraestructura de Clave Publica (PKI), una firma digital indica quien ha firmado un documento electrónico. Sin embargo, la firma puede aún ser repudiada si el documento no incluye una fuente fiable de tiempo.

Un sello de tiempo sobre la firma digital proporciona tiempo preciso de una tercera parte confiable, mostrando cuándo el documento se ha firmado. En ese caso, el documento tiene la propiedad de no repudio.

5.10. El sellado de tiempo en Argentina y el mundo

En esta sección se presenta brevemente el grado de avance del sellado de tiempo en nuestro país y en algunos países del mundo con el fin de graficar la importante evolución que este tema está teniendo.

5.10.1. Brasil

Las normativas reguladoras de la generación de sellos de tiempo se encuentran bajo el ámbito del Instituto Brasileño de Información Nacional de Tecnología de la Información del Gobierno Federal de Brasil (ITI), un ente autárquico vinculado a la Presidencia de la República.

Los certificados utilizados para firmar sellos de tiempo deben ser emitidos por una entidad certificadora que haya obtenido la licencia dentro de la Infraestructura de Clave

Pública del Estado. Sin embargo, los procedimientos para el uso de los sellos de tiempo en Brasil se encuentran en proceso de elaboración por el Comité Gestor del ITI. La hora oficial de este país es emitida por el Observatorio Naval.

5.10.2. Italia

El servicio es brindado por varias empresas privadas entre las cuales se encuentran IT Telecom Italia, Consiglio Nazionale del Natariato, Trust Italia S.p.A., Actalis S.p.A. La normativa reguladora de este servicio se sustenta en el decreto DPCM 080299. La fuente de hora oficial en Italia es brindada por el INRIM (Instituto Nazionale di Ricerca Metrológica).

5.10.3. España

Varias entidades se encuentran inscriptas como Autoridades de Sello de Tiempo en el Ministerio de Industria de este país. Entre ellas se encuentran la Fábrica Nacional de Moneda Timbre, Firmaprofesional, Camerfirma y la Autoritat de Certificación de la Comunitat Valenciana. La fuente de hora oficial es el Real Observatorio de la Armada.

5.10.4. Argentina

La ley 25.506 de Firma Digital y sus normas complementarias no mencionan en forma expresa la existencia de una Autoridad de Sellado de Tiempo.

Sin embargo, el artículo 17, hace mención a la posibilidad de que se brinden otros servicios de certificación relacionados con la firma digital, entre los cuales se puede enmarcar al sellado de tiempo. En la misma tónica, el Decreto 2628/2002, decreto reglamentario de la ley anteriormente mencionada, establece que la Secretaría de Gabinete y Gestión Pública deberá establecer, entre otros aspectos, "las condiciones de prestación de otros servicios relacionados con la firma digital". Esto hace suponer que en el futuro se dictarán reglamentaciones aplicables a este servicio.

Hasta la fecha, no se han detectado empresas u organismos públicos que brinden este servicio.

5.11. Comparación de estándares aplicados al sellado de tiempo

Para dar fin al marco teórico que antecede a la implementación del prototipo de una Autoridad de Sellado de tiempo utilizando software libre, se presenta una tabla comparativa que forma parte de un paper presentado por Jeff Stampleton, quien ha trabajado durante 20 años en varios grupos de trabajo de ISO y en particular en más de 20 estándares relacionados con seguridad entre los cuales se encuentran los relativos al sellado de tiempo digital.

El cuadro 5.1[36] compara los estándares ISO 18014, ANSI X9.95 y la RFC 3161 en varios aspectos: roles, requerimientos, objetos participantes, métodos de sellado de tiempo, controles de auditoría y políticas y procedimientos.

Roles y responsabilidades	**ANS X9.95**	**ISO 18014**	**RFC 3161**
Entidad fuente de tiempo	TSE	-	-
Autoridad de sellado de tiempo	TSA	TSA	TSA
Solicitud de sellado de tiempo	Requirente	Requirente	-
Verificador de sello de tiempo	Verificador	Verificador	-
Req. técnicos y operativos	**ANS X9.95**	**ISO 18014**	**RFC 3161**
Cantidad	Más de 150	22	9
Objetos	**ANS X9.95**	**ISO 18014**	**RFC 3161**
Informe de Calibración de tiempo	ASN.1 y XML		
Requerimientos de sello de tiempo	ASN.1 y XML	ASN.1	ASN.1
Respuesta de sello de tiempo	ASN.1 y XML	ASN.1	ASN.1
Token de sello de tiempo	ASN.1 y XML	ASN.1	ASN.1
Requerimiento de verificación	ASN.1 y XML	ASN.1	-
Respuesta de verificación	ASN.1 y XML	ASN.1	–
Registro de auditoría	Descripto	-	Mencionado
Métodos de sello de tiempo	**ANS X9.95**	**ISO 18014**	**RFC 3161**
Firma digital	Firma digital	Firma digital	Firma digital
MAC	MAC	MAC	-
Tokens linkeados	tokens linkeados	tokens linkeados	-
Archivo	Eliminado	Archivo	
Llave transitoria	Llave “trasient”	-	-
Políticas y procedimientos	**ANS X9.95**	**ISO 18014**	**RFC 3161**
Política de sello de tiempo	22 ejemplos	-	-
Procedimientos de sello de tiempo	Procedimientos modelos	-	-
Objetivos de control de auditoría	**ANS X9.95**	**ISO 18014**	**RFC 3161**
Control de IT	Si	-	-
Control de gestión de claves	Si	-	-
Control de sellos de tiempo	Si	-	-

Cuadro 5.1: Tabla comparativa de las normas existentes

Capítulo 6

Esquemas de sellado de tiempo

En este capítulo se explicarán diversos esquemas de sellado de tiempo que han sido propuestos a los largo del tiempo. Los diversos estándares involucrados en la temática de sellado de tiempo hacen referencia a los mismos y cada uno de ellos permite la aplicación de algunos de los mecanismos existentes.

A continuación se presenta una tabla comparativa ilustrando la relación entre las normas y los modelos de sellados.

Esquema	RFC 3161	ANSI X9.95	ISO/IEC 18014
PKI	Sí	Sí	Sí
Encadenamiento	No	Sí	Sí
MAC	No	Sí	No
Clave transitoria	No	Sí	No
Encadenamiento y firma	No	Sí	No

Cuadro 6.1: Relación entre los mecanismos de sellado de tiempo y la aplicación en los estándares

6.1. Mecanismos téoricos

En la literatura[37][38][39][40][41], los esquemas de sellado de tiempo son clasificados en tres tipos: simple, encadenado y distribuido.

En el *esquema simple*, cada sello nuevo es generado por una Autoridad de Sellado de Tiempo, sin utilizar datos relacionados con otros sellos de tiempo emitidos anteriormente. La principal debilidad de este esquema es que se debe confiar de manera incondicional en el emisor. Si la TSA altera en forma fraudulenta el parámetro de tiempo en un determinado sello, no es posible detectar dicha modificación. Además, si ocurre un robo de la clave privada de la TSA, el tiempo en los sellos de tiempo puede ser falsificado a voluntad. Hasta hace algunos años atrás, solamente era conocido el método de sellado de tiempo basado en el uso de Autoridad de Sellado de Tiempo. De esa forma, las aplicaciones que necesitaban un sellado digital de tiempo no tenían otra alternativa que recurrir a las Autoridades de Sellado de Tiempo, en las cuales debían confiar plenamente. A comienzos de la década del 90, una publicación de Haber

y Stornetta[38] mostró que la confianza en una TSA podía ser mejorada mediante el uso de esquemas de encadenamiento o esquemas distribuidos. A partir de dicho escrito, muchas publicaciones surgieron durante estos últimos años con el objetivo de mejorar los esquemas originales.

La idea principal detrás del *esquema de encadenamiento* es generar un sello de tiempo que involucre datos incluidos en otros sellos de tiempo generados previamente. Una cadena de sellos de tiempo puede ser construida, utilizando funciones de hash de una vía. Si un emisor está dispuesto a alterar o falsificar un sello de tiempo determinado, tiene que alterar indefectiblemente todos los sellos de tiempo relacionados. Por esta razón, es más difícil para un emisor manipular un sello de tiempo en un esquema de encadenamiento que en un esquema simple.

Por ultimo, aparece el *esquema distribuido* donde muchos emisores de manera independiente generan un sello de tiempo siguiendo un esquema simple, en el que cada uno utiliza su propia clave y su propio origen de tiempo. El conjunto de emisores designados para firmar un sello de tiempo es elegido en forma aleatoria, mediante una técnica que impide determinar a priori quienes serán los firmantes de un sello de tiempo determinado. Este esquema confía en la dificultad que tiene el solicitante del sello de tiempo para complotar con un gran número de emisores con el objetivo de concretar el proceso de sellado. Como contrapartida, la necesidad de un gran número de emisores independientes, hace que el esquema distribuido sea prácticamente inviable en los escenarios planteados del mundo real.

En conclusión, los algoritmos utilizados en la mayoría de los casos prácticos implementados utilizan esquemas de encadenamiento, debido a que los mismos proveen un buen balance en términos seguridad y performance.

El principal objetivo del sellado de tiempo es la autenticación temporal, es decir la posibilidad de probar que un cierto documento ha sido creado antes de un determinado momento. Un sistema de sellado de tiempo puede ser pensado como un conjunto de supervisores, de un servidor de sellado de tiempo y un tupla de tres protocolos (S, V, A). El protocolo de sellado S permite a cada participante enviar un mensaje. El protocolo de verificación V es utilizado por el supervisor para verificar el tiempo en el que se ha generado una estampa de tiempo o el orden relativo de todos los sellos generados. El protocolo de auditoría A es utilizado por el supervisor para verificar si un servidor de sellado de tiempo lleva a cabo sus funciones en forma correcta.

6.1.1. Esquema simple

Como se ha explicado, este esquema confía solamente en la TSA quien está a cargo de certificar el tiempo mediante la firma del sello. De esta forma, el protocolo de sellado S consiste en una simple firma digital del mensaje del solicitante. Más precisamente, un sello de tiempo es realizado de esta forma:

- El cliente envía un docuento x o un hash del mismo a la TSA.
- La TSA agrega el tiempo actual t y el identificador de la TSA *ID* y firma el documento compuesto *(ID,t,X)*.
- La TSA retorna los dos valores t y $s=sig_{TSA}(ID,t,X)$ al cliente.

El protocolo de verificación V chequea la integridad de la firma digital y establece una orden relativo temporal entre dos sellos comparando los tiempo absolutos. El protocolo de auditoría A puede solamente revelar un comportamiento inadecuado de la TSA enviando requerimientos de sellado de tiempo de prueba y chequeando la correctitud de los valores de tiempo certificados. Aunque es importante comprender que esta técnica de chequeo no tiene la potestad de chequear un sello de tiempo genérico previamente enviado y firmado.

Pero es importante recalcar nuevamente que si la clave privada de la TSA es comprometida, no hay forma de distinguir entre un sello genuino de uno fraudulento.

6.1.2. Esquema de encadenamiento

El esquema de encadenamiento[39] fue introducido para superar los inconvenientes del esquema de sellado anterior. Aunque la creación del dato digital es observable en el mundo físico, el momento de su creación no puede ser determinado por la observación de los datos en sí. Sin embargo, es posible chequear el orden temporal relativo de la creación de esos elementos mediante el uso de funciones de una vía definiendo la flecha del tiempo.

Todos los esquemas propuestos de encadenamiento de sellos generan dependencias mediante el uso de las funciones de hash de una vía, las cuales fueron explicadas en un capítulo anterior.

En este esquema, la función de hash es usada principalmente para producir dependencias temporales entre los sellos emitidos, basándose en la siguiente consideración: Si h es una función de hash, y el valor $h(x)$ y x son conocidos para el supervisor P en el momento t, entonces alguien usó x para computar $h(x)$ en un momento previo a t.

El protocolo S utilizado por una TSA que sigue un esquema de encadenamiento puede ser resumido de la siguiente forma:

- El cliente envía el documento o su valor de resumen a la TSA.
- La TSA combina los requerimientos de clientes individuales que arriban en una ventana de tiempo determinada junto a ciertos valores relacionados a sellos emitidos en el pasado.
- La TSA firma un documento compuesto, el cual es una función de un número de sellos emitidos.
- La TSA retorna el sello de tiempo al solicitante.

Obviamente, con este esquema es más complicado producir sellos de tiempo adulterados porque alterar un simple sello significa alterar todas las dependencias verificables. Por otra parte, la necesidad de una TSA de confianza puede reducirse en gran medida con los esquemas de encadenamiento a través de la publicación periódica de los valores utilizados para crear las dependencias. Actualmente si uno puede demostrar la dependencia de un sello en un dato ampliamente aceptado, la TSA no se encuentra más involucrada en el proceso de verificación. El protocolo de verificación V puede entonces seguir el camino de dependencias desde el sello de tiempo en cuestión hasta el dato ampliamente reconocido. El protocolo de auditoría A tiene como objetivo chequear la

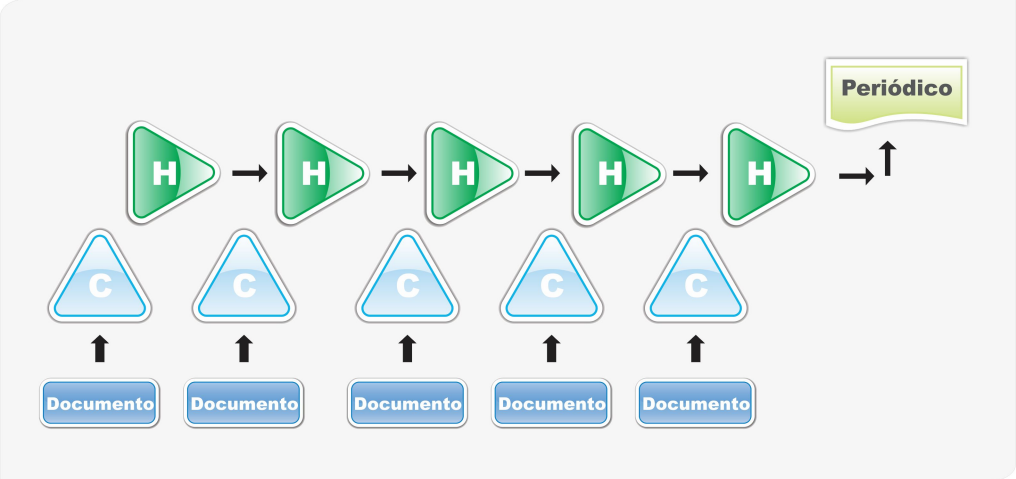

Figura 6.1: Esquema de encadenamiento

integridad de los esquemas de encadenamiento utilizando los datos publicados. La auditoría puede ser realizada en cualquier momento y permite en el momento de verificación saber no solamente si un proceso de encadenamiento fue alterado sino conocer cuándo fue realizada tal acción y de esa manera distinguir sellos genuinos de sellos que fueron alterados.

6.1.3. Esquema distribuido

Un enfoque alternativo propone basarse en la confianza distribuida. En este esquema no es necesario utilizar un servidor de sellado de tiempo central. Se supone que hay un esquema de firma seguro de forma que cada usuario puede firmar mensajes y que un generador estándar pseudo aleatorio G está disponible para todos los usuarios. Este generador es un algoritmo que expande las semillas de entrada en secuencias de salida que son impredecibles.

El protocolo S seguido por un usuario en un esquema distribuido puede ser resumido de la siguiente forma:

- Dado de valor de hash de un documento a ser sellado, el generador pseudoaleatorio es utilizado para generar una k-tupla de números identificadores de clientes: *G(y)* = *(*ID_1*,*ID_2*,....,*ID_k*)*. Cada cliente correspondiente al ID1,ID2,....,IDk recibe la solicitud de sellado de tiempo junto con y y el identificador.

- Cada cliente designado por $G(y)$ firma el mensaje *(t,ID,y)* incluyendo el tiempo t y retornando el mensaje firmado. El sello de tiempo consiste de *[(y,ID),(*s_1*,....,*s_k*)]*.

Como en el esquema simple, el protocolo de verificación V chequea la integridad de la firma digital de todas las respuestas, mientras que el protocolo de auditoría A puede detectar infracciones en el sistema enviando requerimientos de sellados de tiempo de prueba y chequeando la correctitud y consistencia de todos los valores de tiempo retornados. De igual manera, esta técnica tampoco puede chequear un sello de tiempo previamente emitido y correctamente firmado. La fuerza de este sistema se basa en que el remitente debería actuar en complicidad con todos los clientes seleccionados.

Las propiedades del generador pseudoaleatorio G asegura que es computacionalmente imposible encontrar un documento. Para obtener una sello fraudulento, el remitente debería coordinar con un número k de clientes aleatorios. El sistema distribuido presenta una serie de incovenientes. En primer lugar, el conjunto de posibles clientes debe ser establecido a priori y el generador pseudoaleatorio debería determinar solamente clientes existentes. Aún si esta condición es asegurada, la disponibilidad de los clientes elegidos al azar pueden determinar en forma sencilla una debilidad en el sistema. Más aún, el ancho de banda requerido para un sello de tiempo puede ser k veces más grande que el necesario en el esquema simple. Otros problemas se relacionan con la sincronización de muchas fuentes diferentes de tiempo y con la definición legal de la responsabilidad por el tiempo certificado. De hecho, el método distribuido, propuesto junto con el método de encadenamiento en el año 1991, no se ha estudiado en profundidad hasta el momento.

Este método requiere el uso de una PKI para poder autenticar la clave pública de la TSA y sus usos asociados.

6.2. Otros mecanismos

En este sección se explicarán otros mecanismos de sellado de tiempo que se encuentran mencionados en las diversas normativas relacionadas con el sellado de tiempo.

6.2.1. Método de MAC

El código de autenticación de mensaje(MAC) es un mecanismo utilizado para autenticar un mensaje. El algoritmo de este método, toma como entrada una clave secreta y un mensaje de longitud arbitraria, y devuelve un MAC. Este valor, es utilizado para garantizar la integridad y la autenticidad de los datos, permitiendo a los verificadores, que deben poseer la clave secreta utilizada, detectar cualquier cambio en el contenido del mensaje.

Este mecanismo difiere de la firma digital en que tanto la generación como la verificación de los datos es realizada con la misma clave secreta. Esto implica que el emisor y el receptor de un mensaje deben acordar la clave previamente al inicio de la transmisión de los mensajes.

En el marco de un servicio de sellado de tiempo, este método es el cuál la TSA utiliza una clave simétrica para ligar criptográficamente el token con un código de mensaje de autenticación. La verificación del token es realizada mediante el chequeo del MAC por parte del verificador. La clave secreta utilizada para computar el MAC debe ser mantenida en forma segura y debe estar disponible en el momento que se requiera realizar la verificación. La clave secreta puede ser específica para un token en particular o puede ser utilizada para un rango determinado de tokens emitidos.

6.2.2. Metodo de clave transitoria

La criptografía de clave transitoria es una variante de la criptografía de clave pública donde el par de claves son generados y asignados a breves lapsos de tiempo en lugar de ser asignados a personas, servidores u organizaciones. En un sistema de clave transitoria, las claves privadas son utilizadas y luego destruidas. Los datos encriptados con una clave

privada asociada a un intervalo de tiempo determinado pueden ser vinculados en forma irrefutable a ese intervalo, haciendo que este mecanismo sea muy útil en el sellado digital de tiempo.

En este sistema, la fuente de tiempo utilizada para la generación de las claves transitorias, debe ser confiable por todos las entidades que utilicen el servicio. Cuando un intervalo de tiempo expira, un nuevo par de claves es generado, y la clave privada del intervalo previo es utilizado para firmar la nueva clave pública que identifica al nuevo intervalo. En el instante posterior a la firma, la clave privada del intervalo expirado es destruida.

Para verificar un dato que ha sido sellado tiempo atrás, el verificador debe utilizar la clave pública correspondiente al intervalo de tiempo en el cual fue generado el sello. La clave pública es aplicada a la firma digital para poder obtener el hash del dato original, el cual es comparado con el hash que se encuentra almacenado en los repositorios. Si es posible desencriptar la firma utilizando la clave pública de un intervalo en particular, entonces se puede afirmar en forma inequívoca que el dato fue sellado en dicho período.

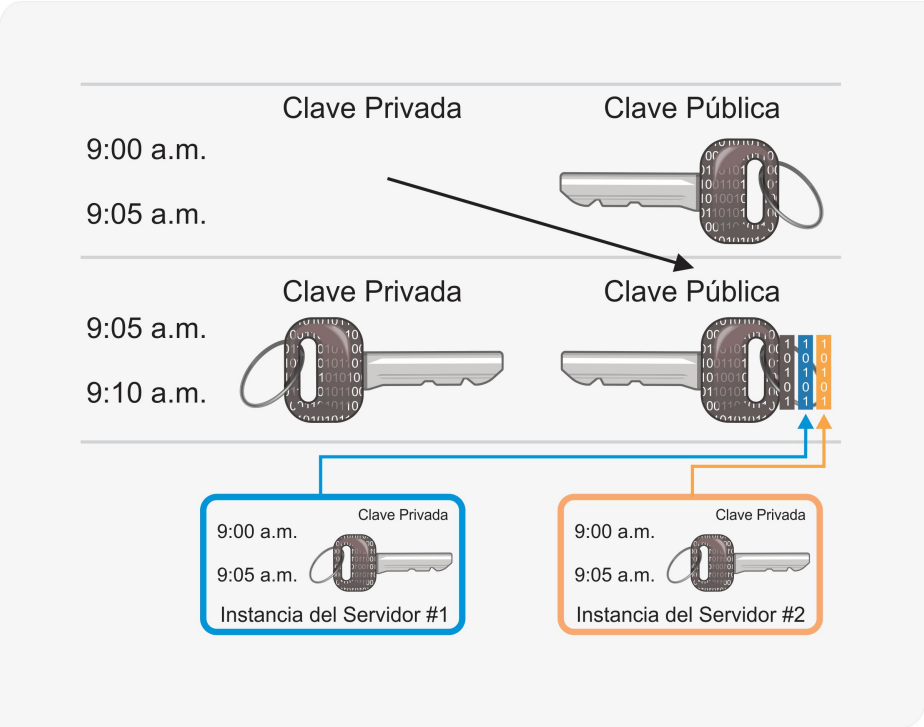

Figura 6.2: Método de clave transitoria

Capítulo 7

Mecanismos de transporte

En la RFC 3161 no se define un mecanismo de transporte mandatorio para los mensajes de sellado de tiempo, es decir la misma no establece una regla de cómo los mensajes deben ser transmitidos entre un cliente y un servidor. El IETF describe cuatro tipos de mecanismos de transporte que pueden utilizados opcionalmente. Además indica claramente que en el futuro pueden ser definidos mecanismos adicionales.

Por su parte, la ETSI obliga a tener un protocolo en línea y uno de almacenamiento y envío para cada TSA. Dentro de los cuatro protocolos mencionados en la RFC 3161, esta norma sugiere que debería utilizarse el mecanismo basado en HTTP.

Particularmente, la ISO no realiza ningún comentario o recomendación en relación al mecanismo de transporte a utilizar.

En las siguientes secciones de este capítulo se explicarán los diversos mecanismos de transporte planteados por la RFC 3161. En primer término se presentará el protocolo basado en e-mail, seguido del protocolo basado en sockets. Posteriormente se explicará el protocolo basado en archivos y para finalizar, se explicará con mayor detalle el protocolo basado en HTTP, el cual es el mayormente implementado por las diversas organizaciones que brindan servicios de sellado de tiempo.

7.1. Protocolo basado en e-mail

Se definen los siguientes objetos MIME para la petición y la respuesta, respectivamente, de un sello de tiempo:

Content-Type: application/timestamp-query Content-Transfer-Encoding: base64 «the ASN.1 DER-encoded Time-Stmp message, base64-encoded»
Content-Type: application/timestamp-reply Content-Transfer-Encoding: base64 «the ASN.1 DER-encoded Time-Stmp message, base64-encoded»

Estos objetos MIME pueden ser enviados utilizando procesos MIME, y proporciona un mecanismo sencillo de transporte para los mensajes de sellado de tiempo.

Los tipos MIME *application/timestamp-query* y *application/timestamp-reply*, podrían incluir opcionalmente los parámetros *name* y *filename*. De esta manera, se conserva el

tipo de información cuando se guardan como archivos. Cuando se incluyen estos dos parámetros, el nombre del archivo debe tener la extensión adecuada:

MIME types	File Extension
application/timestamp-query	.TSQ
application/timestamp-reply	.TSR

Cuadro 7.1: Extensiones MIME

7.2. Protocolo basado en sockets

Este protocolo es adecuado cuando una entidad inicia una transacción y puede conectarse a un puerto para obtener los resultados; requiere que en la TSA haya un proceso en un puerto determinado, el 318, escuchando para poder aceptar los mensajes. El resultado obtenido por la TSA podría ser el mensaje de respuesta o el número de otro puerto mediante el cuál podrá recuperar el resultado.

7.3. Protocolo basado en archivos

La RFC 3161 explica que un archivo que contenga un mensaje de sellado de tiempo, debe tener solamente el mensaje codificado en formato DER, sin ningún otro dato adicional. Estos archivos pueden ser utilizados para ser transportados mediante FTP, para citar un posible ejemplo.

Una petición de sellado de tiempo debe ser contenida en un archivo con extensión **.tsq** y la respuesta debe estar contenida en un archivo cuya extensión sea **.tsr**.

7.4. Protocolo basado en HTTP

En esta sección se van a explicar los mensajes de petición y respuesta utilizando el protocolo HTTP, según lo especificado en la RFC 3161. Se detallarán los mensajes de solicitud de sello de tiempo por parte del cliente y la posterior respuesta por parte de la TSA.

7.4.1. Timestamping Request

Este mensaje lo utiliza la entidad que quiere un sello de tiempo (solicitante) para acceder al servicio que ofrece una TSA. Tiene el siguiente formato:

```
TimeStampReq ::= SEQUENCE {
   version                INTEGER { v1(1) },
   messageImprint         MessageImprint,
   reqPolicy              TSAPolicyId              OPTIONAL,
   nonce                  INTEGER                  OPTIONAL,
   certReq                BOOLEAN                  DEFAULT FALSE,
   extensions             [0] IMPLICIT Extensions OPTIONAL
}
```

Campo	Descripción
Version	Versión de la petición TimeStamp (v1).
messageImprint	OID del algoritmo hash y el valor del hash de los datos.
Nonce	Si se incluye el nonce permite al cliente comprobar el retardo en la respuesta cuando no se dispone de reloj local. La respuesta debe contener este mismo número o se rechazará. El nonce es un número aleatorio con una elevada probabilidad de que el cliente lo genere una única vez (entero de 64 bits).
CertReq	Si el campo certReq está presente y con valor true, la clave pública de la TSA debe estar referenciada por el identificador ESSCertID dentro de un atributo SigningCertificate de la estructura SignedData en la respuesta. Ese campo además puede contener otros certificados. Si falta el campo certReq o tiene valor false entonces, el campo SigningCertificate de la estructura SignedData no debe aparecer en la respuesta.
extensions	Es una forma de permitir añadir nuevos campos en el futuro. Si se incluye algún campo de extensión que la TSA no reconozca, esta devolverá un mensaje de error de extensión no aceptada (unacceptedExtension).

```
MessageImprint ::= SEQUENCE {
        hashAlgorithm          AlgorithmIdentifier,
        hashedMessage          OCTET STRING
}
```

Campo	Descripción
hashAlgorithm	OID del algoritmo hash: El algoritmo de hash indicado en hashAlgorithm debería ser uno conocido por la TSA. También comprobará que sea suficientemente fuerte. Si la TSA no reconoce el algoritmo usado, entonces la TSA denegará el servicio al cliente devolviendo un pkiStatusInfo de 'bad_alg'.
hashedMessage	Este campo contiene el hash de los datos que se quiere sellar. La longitud del hash tiene que coincidir con la longitud de hash del algoritmo utilizado.

El mensaje Timestamp Request no identifica al cliente, y esta información no es validada por la TSA.

7.4.2. Timestamping Response

Es la respuesta que la TSA da a una mensaje time stamp request. Tiene la siguiente representación:

```
TimeStampResp ::= SEQUENCE {
```

```
      status                  PKIStatusInfo,
      timeStampToken          TimeStampToken OPTIONAL
}
```

Campo	Descripción
Status	Estado de la respuesta.
timeStampToken	Este campo que contiene la marca de tiempo generado. Es una estructura ContentInfo que encapsula información firmada en una estructura TSTInfo. Está definida en la RFC 2630.

```
PKIStatusInfo ::= SEQUENCE {
      status        PKIStatus,
      statusString PKIFreeText     OPTIONAL,
      failInfo      PKIFailureInfo OPTIONAL
}
```

```
TSTInfo ::= SEQUENCE {
   version              INTEGER { v1(1) },
   policy               TSAPolicyId,
   messageImprint       MessageImprint,
   serialNumber         INTEGER,
   genTime              GeneralizedTime,
   accuracy             Accuracy                 OPTIONAL,
   ordering             BOOLEAN             DEFAULT FALSE,
   nonce                INTEGER                  OPTIONAL,
   tsa                  [0] GeneralName          OPTIONAL,
   extensions           [1] IMPLICIT Extensions   OPTIONAL
}
```

Campo	Descripción
Status	Estado de la respuesta. ▪ **granted(0)**: Marca de tiempo presente. ▪ **grantedWithMods(1)**: Marca de tiempo presente con modificaciones. ▪ **rejection(2)**: Petición rechazada. ▪ **waiting(3)**: Esperando. ▪ **revocationWarning(4)** : Advertencia de revocación inminente. ▪ **revocationNotification(5)**: Notificación de revocación.
statusString	Puede utilizarse para indicar eventos de error.
failInfo	Causas del fallo. ▪ **badAlg(0)**: Identificador de algoritmo no soportado. ▪ **badRequest(2)**: Transacción no permitida o soportada. ▪ **badDataFormat(5)**: Datos enviados con formato incorrecto. ▪ **timeNotAvailable(14)**: Origen de tiempo no disponible. ▪ **unacceptedPolicy(15)**: Política solicitada no soportada. ▪ **unacceptedExtension(16)**: Extensión no soportada. ▪ **addInfoNotAvailable(17)**: Información adicional no disponible. ▪ **systemFailure(25)**: Error del sistema.

Campo	Descripción
Version	Versión de la respuesta TimeStamp (v1).
ReqPolicy	OID de la política de la TSA. Indica la política de la TSA bajo la cual se proporciona el sello. Si se ha generado el sello, será igual al del mensaje de petición.
messageImprint	OID del algoritmo hash y el valor del hash de los datos. Debe tener el mismo valor de messageImprint que el campo correspondiente de la petición.
serialNumber	Es un entero asignado por la TSA y debe ser único para cada sello que genere. Por tanto, un sello será identificado por el nombre de la TSA que lo generó y el número de serie asignado. Permite hasta 160 bits.
genTime	Es el instante de tiempo en el que se creó el sello. Tanto ISO como el IETF expresan el instante de tiempo referido a la escala UTC, para evitar confusiones con las horas locales. El formato debe ser el siguiente: **CC YY MM DD hh mm ss Z**. ▪ CC representa el siglo (19-99) ▪ YY representa el año (00-99) ▪ MM representa el mes (01-12) ▪ DD representa el día (01-31) ▪ hh representa la hora (00-23) ▪ mm representa los minutos (00-59) ▪ ss representa los segundos (00-59) ▪ Z viene de zulu, que es como se conoce a la escala UTC
accuracy	Representa la desviación del tiempo UTC contenido en genTime, en los casos que sea necesario, proporciona una precisión incluso de microsegundos.
ordering	Si falta el campo ordering o está presente y tiene valor false, entonces el campo genTime solo indica el momento en el que la marca de tiempo ha sido creada por la TSA. En este caso, el orden de la marcas de tiempo emitidas por una misma TSA o distintas TSAs sólo es posible cuando la diferencia entre el genTime de la primera marca de tiempo es mayor que la suma de las precisiones del genTime de cada marca de tiempo.
nonce	El nonce es un número aleatorio con una elevada probabilidad de que el cliente lo genere una única vez (entero de 64 bits). Debe tener el mismo valor que el campo correspondiente de la petición.
tsa	Identificador de la TSA.
extensions	Es una forma de permitir añadir nuevos campos en el futuro.

Capítulo 8

Componentes

En este capítulo se explicarán las decisiones de diseño y selección de componentes. Para cada uno de los componentes necesarios para implementar una Autoridad de sellado de tiempo, se analizarán las ventajas y las desventajas de las distintas alternativas disponibles teniendo como precondición esencial el requisito de utilizar software libre para toda la solución.

En primer término, se explicarán los distintos criterios de selección de componentes y a continuación para cada uno de ellos se presentarán los productos o soluciones existentes, se las comparará, y se tomará y justificará la decisión tomada.

Los componentes a discutir comprenden el producto de implementación de la RFC 3161, el sistema operativo y el servidor de base de datos para dar soporte de almacenamiento a la información generada en el prototipo.

Seguidamente. se explicarán las cuestiones relativas a la integración del servicio de sellado de tiempo con la Autoridad de Certificación de la Universidad Nacional de La Plata y luego se hará mención a los distintos algoritmos de firma y hashing que serán utilizados en el servicio implementado, teniendo como restricciones principales la normativa existente y el estado del arte en relación a las vulnerabilidades encontradas en los algoritmos criptográficos vigentes.

Para finalizar, se explicitará la precisión de los relojes que serán utilizados por el servicio de sellado de tiempo para determinar la fecha y hora exacta de la operación.

8.1. Criterios de selección de componentes

Para la selección de los distintos componentes del servicio a implementar se tuvieron en cuenta diversos criterios:

- Adecuación a la normativa existente relativa a los servicios de sellado digital de tiempo.
- Licencia del software utilizado.
- Comunidad que utiliza los potenciales productos o tecnologías.
- Documentación existente y facilidad de acceso a la misma.
- Experiencia previa.

8.2. Selección de componentes

8.2.1. Producto de implementación de RFC 3161

OpenTSA

El objetivo del proyecto OpenTSA[42] es desarrollar una autoridad de sellado de tiempo sin costo alguno y de código abierto. Dicha aplicación es compatible con la RFC 3161 y comprende lo siguiente:

- **Integración con OpenSSL[43]**: La creación de las peticiones de sellado de tiempo, la generación de las respuestas y la verificación de las mismas, son funcionalidades que fueron implementadas como una extensión para la última versión estable de OpenSSL. Este parche agrega los comandos adicionales para que las operaciones de sellado de tiempo se puedan llevar a cabo. Actualmente, el parche y el cliente de sellado de tiempo han sido integrados en la versión oficial y están disponibles a partir de la versión 1.0, la cual ha sido liberada y se encuentra estable.

- **Módulo para Apache**: Este paquete es un módulo para el servidor HTTP Apache[44]. Utilizando la funcionalidad de OpenSSL, este módulo funciona como un servidor que cumple con lo especificado en la RFC 3161 y utiliza tanto HTTP como HTTPS como protocolos de transporte. Trabaja con una base de datos relacional, específicamente con PostgreSQL, MySQL o Firebird como variantes posibles.

- **Cliente de Sellado de tiempo**: Junto con OpenSSL se distribuye una serie de comandos que permiten la creación y el envío de requerimientos de sellado de tiempo sobre HTTP o HTTPS y a su vez provee la funcionalidad para verificar las respuestas recibidas.

La licencia de OpenTSA es la misma que posee OpenSSL. La licencia de de este último es una combinación de dos licencias: la propia licencia de OpenSSL y la de SSLeay. Tienen que cumplirse ambas licencias al utilizar OpenSSL. La combinación tiene como resultado una licencia de software libre incompatible con la GNU GPL. Esto se debe, principalmente, a que tiene una cláusula de publicidad que requiere mencionar OpenSSL en el momento de ser utilizada.

La última versión del módulo para Apache data del año 2006, pero la integración del módulo de sellado de tiempo en las próximas versiones de OpenSSL hablan de la vigencia de una solución. Actualmente Zoltan Glozik, el mentor principal del proyecto, contribuye en la documentación del módulo dentro de OpenSSL.

OpenEvidence

Financiado por la comunidad europea, OpenEvidence[45] - parte del Grupo de Proyecto Europeo FP5 - es un framework de código abierto para la certificación, sellado temporal y archivo de datos que brinda tecnología para la creación de evidencias, validación y protección a largo plazo de documentos electrónico.

Como partícipe del proyecto, la empresa C&A contribuyó con una implementación que respeta el estándar de la RFC 3161. Provee como mecanismos de transporte HTTP,

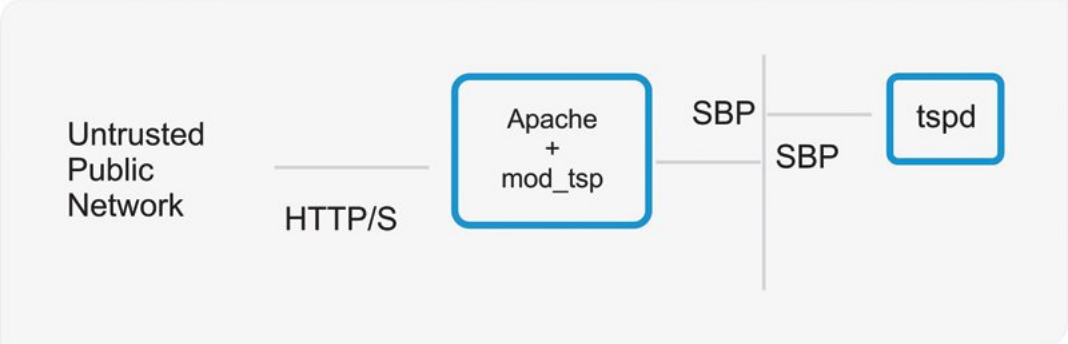

Figura 8.1: Arquitectura de OpenEvidence

HTTPS y el protocolo basado en sockets. La funcionalidad es provista a través dos módulos de software separados: el módulo de Apache y un demonio de UNIX.

- **Módulo Mod_tsp**: Este programa es un módulo para el servidor de HTTP Apache, y se encarga de convertir las peticiones HTTP o HTTPS al protocolo basado en sockets y las respuestas del protocolo basado en sockets a HTTP o HTTPS. Puede ser utilizado como interfaz del sistema en conjunto con una parte trasera o backend que complete la funcionalidad restante. Por sí solo, no provee ninguna funcionalidad de sellado de tiempo.
- **Demomio tspd**: Este programa corre como un demonio del sistema operativo e implementa las funcionalidades principales de un servicio de sellado de tiempo. Se comunica con el otro módulo utilizando una versión minimizada del protocolo basado en sockets, aceptado por la normativa existente. Ante la falta de mecanismos de seguridad que este protocolo posee, los desarrolladores recomiendan encarecidamente ubicar este componente del sistema en un red protegida, colocando solamente el servidor Web como punto de entrada al sistema.

El proyecto se dio de baja en el mes de abril del año 2004[46] y la documentación relacionada con este producto es escasa. Diversos empresas han implementado soluciones privativas utilizando como base de su desarrollo el framework de OpenEvidence.

Desarrollo propio

Otra posible alternativa para implementar un servicio de sellado digital de tiempo es desarrollar una solución que cumpla con las normativas existentes, en particular lo especificado en la RFC 3161. Uno de los objetivos planteados en la tesis pone énfasis en la implementación de una infraestructura de autoridad de sellado de tiempo para brindar un servicio 7X24; el desarrollo de una aplicación nueva se encuentra fuera del alcance de este trabajo.

Existen aplicaciones y librerías públicas que implementan el protocolo en diversos lenguajes de programación:

- **BouncyCastle**[**47**]: Conjunto de librerías criptográficas que implementan el protocolo TSP en los lenguajes Java y C#.

- **OpenTSA**: Es una ampliación de la librería criptográfica OpenSSL que implementa el protocolo TSP en lenguaje C.
- **Digistamp**[48]: Toolkit basado en la librería criptográfica CryptoAPI de Microsoft que implementa el protocolo TSP en Visual Basic.
- **IAIK**: Incluye librerías criptográficas en Java que implementan el protocolo TSP. Estas librerías son gratuitas únicamente para propósitos no comerciales.
- **Adobe Reader**: La aplicación Adobe Reader 8 permite validar sellos de tiempo incluidos en documentos PDF.

Producto elegido

Luego de haber explorado las opciones existentes para poner en marcha el servicio, he decidido utilizar el producto de OpenTSA por diversas razones. El principal motivo se basa en la arquitectura de la aplicación. Su desarrollo en forma modular, y su integración con herramientas que están en plena vigencia como son OpenSSL y Apache, permiten que este producto pueda ser utilizado independientemente de las actualizaciones existentes en las sucesivas versiones de ambos productos. Otro aspecto que me lleva a tomar dicha decisión es la existencia de documentación y una comunidad que respalda de alguna forma la continuidad del proyecto a pesar de que el módulo no sufre actualizaciones desde hace un tiempo.

En relación a OpenEvidence, la ausencia de una buena documentación me ha impedido configurar en forma correcta el producto y no he encontrado una comunidad importante que utilice el producto de código abierto; solamente he hallado referencias a empresas que han tomado este proyecto como disparador de soluciones comerciales que se ofrecen principalmente en diversos países europeos.

8.2.2. Sistema operativo

Al momento de seleccionar el sistema operativo para montar el prototipo, la premisa planteada de utilizar software libre, reduce el espectro de productos a seleccionar. Por lo tanto todos los sistemas pertenecientes a la empresa Microsoft, varios sistemas UNIX propietarios como AIX de IBM[49] y Solaris[50] de Sun Microsystems, adquirida recientemente por Oracle, y algunas distribuciones Linux como SuSE Enterprise, quedan relegadas en el proceso de selección.

Por lo tanto, a la hora de optar por alguna distribución de sistema operativo, los sistemas operativos que cumplen con la licencia de software libre se limitan a una distribución GNU/Linux o alguna distribución derivada de BSD.

BSD

El Berkeley Software Distribution es un sistema operativo derivado de UNIX que fue desarrollado y distribuido por un grupo de investigación de la Universidad de California entre los años 1977 y 1995. Históricamente, BSD fue considerado como una rama de UNIX - razón por la cuál se lo denomina comúnmente BSD UNIX - porque compartía aspectos de codificación y diseño con el UNIX de AT&T, pero a lo largo de se evolución

fue transformándose en una alternativa libre a los sistemas operativos existentes en esos tiempos.

A través del desarrollo de las distintas versiones, BSD ha hecho grandes contribuciones en el campo de los sistemas operativos en general, entre las cuales se destacan:

- el manejo de memoria virtual paginado por demanda.
- el control de trabajos.
- el Fast FileSystem.
- el protocolo TCP/IP (casi todas las implementaciones de TCP derivan de la de 4.4 BSD-Lite).

La última distribución creada por Berkeley fue el BSD 4.4-Lite Release 2, lanzado en 1995, después de que el grupo de investigación de la Universidad fuera disuelto. Desde ese momento han aparecido muchas distribuciones basadas en BSD 4.4, tales como FreeBSD[51], OpenBSD[52] y NetBSD[53].

GNU/Linux

GNU/Linux es una implementación de libre distribución UNIX para computadoras personales (PC), servidores, y estaciones de trabajo. En la actualidad es soportado por una gran cantidad de procesadores diferentes. Entre sus principales características se puede mencionar que es multiusuario, multitarea, multiplataforma y multiprocesador. Hacia la década del 80, la mayoría del software se había vuelto propietario; tenía dueños que prohibían y evitaban la cooperación entre los usuarios. Esto hizo que en 1983, Richard Stallman concibiera la Free Software Foundation (Fundación software libre, FSF) y en el marco de la misma el proyecto GNU[54], como una forma de recuperar el espíritu cooperativo de los primeros días de la computación, y posibilitar nuevamente la cooperación sacando los obstáculos impuestos por los dueños del software propietario. El objetivo principal de GNU es desarrollar un sistema operativo y un conjunto de aplicaciones para correr sobre el que fueran compatibles con UNIX pero distribuidas con licencia de software libre. En 1990, se habían escrito la mayoría de los componentes principales del sistema operativo excepto uno: el kernel o núcleo. Para ese entonces, Linux comenzó como proyecto personal del entonces estudiante Linus Torvalds; combinando Linux con el resto del sistema GNU se llegó a la meta inicial de un sistema operativo libre: El sistema GNU basado en Linux. En el ambiente GNU/Linux hay un concepto que es la distribución. Una distribución es una recopilación de programas y archivos, organizados y preparados para su instalación. Entre las principales distribuciones se encuentran Debian[55], Red Hat, Fedora, Suse, Ubuntu, Gentoo, Slackware y CentOS.

Producto elegido

Al momento de decidir cuál sistema operativo utilizar, la comunidad que utiliza estos productos y la documentación existente no son factores que puedan pesar en el momento de la elección. Esto se debe a que tanto las diversas versiones de BSD y GNU/Linux cuentan con comunidades de miles de usuarios que generan documentación

con gran asiduidad. Por lo tanto la experiencia previa va a ser el elemento que permita optar entre las alternativas disponibles.

Desde el año 2006, he utilizado GNU/Linux en el ámbito de la Facultad, tanto en mi computadora personal como en los servidores que administro. Además, en las diversas cátedras que participo, utilizamos y fomentamos el uso de GNU/Linux para la realización de las distintas actividades prácticas involucradas. Con respecto al uso de alguna distribución derivada de BSD, no he tenido grandes experiencias si la comparo con la cantidad de instalaciones y servicios que he montado sobre GNU/Linux.

Por lo tanto, la experiencia personal hace que termine optando por GNU/Linux como la alternativa más viable. Nuevamente, este factor se convierte en uno de los trascendentales en el momento de optar por Debian GNU/Linux como distribución a utilizar, sumado a la facilidad de uso, la gran comunidad y la documentación existente que terminaron volcando la balanza definitivamente.

8.2.3. Base de datos

Con respecto a la selección del producto de base de datos a utilizar, la decisión de utilizar el producto OpenTSA como servicio de Servicio de sellado de tiempo, restringe los motores de base de datos a utilizar. El producto, según la documentación provista, soporta tres motores de base de datos distintos: MySQL, PostgreSQL y Firebird. Luego de explicar brevemente las características distintivas de cada uno de estos productos, seleccionaré el producto basándome en los criterios de selección mencionados anteriormente.

MySQL

MySQL[56] es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones, según informes del año 2005. MySQL AB —desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009— desarrolla MySQL como software libre en un esquema de licenciamiento dual. Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellos que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Debido a estas características relacionadas con el licenciamiento del producto, la comunidad de MySQL ha creado varios productos derivados del mismo, entre los cuales se pueden mencionar a Drizzle y MariaDB. Estos proyectos han tenido mayor relevancia desde la compra de MySQL por parte de Oracle y a diferencia de MySQL, se distribuyen con licencia GNU GPL para todos los usos.

PostgreSQL

PostgreSQL[57] es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD.

Como muchos otros proyectos de código abierto, y a diferencia de lo que sucede con MySQL, el desarrollo de PostgreSQL no es administrado por una sola empresa sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales que trabajan en el desarrollo del mismo. Dicha comunidad es conocida como PGDG (PostgreSQL Global Development Group).

El proyecto PostgreSQL continúa realizando lanzamientos principales en forma anual y lanzamientos menores de reparación de bugs, todos disponibles bajo la licencia BSD, y basados en contribuciones de proveedores comerciales, empresas aportantes y programadores independientes.

Firebird

Firebird[58] es un sistema de administración de base de datos de código abierto, basado en la versión 6 de Interbase.

A finales de la década de 1990, Borland decidió liberar el código de Interbase. Diversos integrantes de la organización crearon una nueva empresa denominada IBPhoenix, y junto a otros desarrolladores independientes, crearon la rama que ahora es conocida como Firebird. Más tarde, Borland volvió a privatizar Interbase y a comercializar sus licencias. Sin embargo, Firebird continúa siendo un proyecto de código abierto y no tiene licencias duales como lo tiene MySQL; es posible utilizarlo en aplicaciones comerciales como aplicaciones de código abierto.

En relación a las características, cuenta con las funcionalidades tradicionales de un motor de base de datos actual, lo que permite ser utilizado en gran cantidad de aplicaciones de diversa índole.

Producto elegido

Al momento de elegir un producto como motor de base de datos del prototipo de sellado de tiempo, el licenciamiento es un punto a tener en cuenta, y las diversas licencias existentes en MySQL, junto con la aparición de Oracle como propietario de dicho producto hacen éste sea descartado de plano. Es importante tener en cuenta que Oracle no garantiza que las futuras versiones de MySQL tengan la licencia GNU GPL que existe hasta el momento, y no es posible predecir lo que la empresa pueda decidir en los próximos meses.

Entonces, resultan dos las alternativas disponibles: Firebird o PostgreSQL.

Al momento de armar un infraestructura para soportar un servicio que requiere estar disponible 7X24, con una potencial gran cantidad de usuarios concurrentes accediendo al servicio, hay varias características deseables en un motor de base de datos: la replicación y el clustering tanto para balanceo de carga como para tolerancia a fallos. Estas características no están disponibles en Firebird mientras que PostgreSQL las ofrece como una de sus características distintivas.

Además, en relación a la comunidad y la documentación existente, PostgreSQL cuenta con una gran cantidad de documentación en línea, foros, listas de correo, canales de chat, libros y una gran comunidad que contribuye a diario en el proyecto. Adicionalmente, gran cantidad de empresas reconocidas, utilizan dicho producto. Por su parte, Firebird posee una menor cantidad de usuarios, una pobre documentación y no cuenta con grandes sponsors, aspectos que justifican en parte el lento crecimiento del proyecto y el espaciado lanzamiento de nuevas versiones.

Con respecto al licenciamiento, los productos en discusión cuentan con licencias que se adaptan a los requerimientos de la implementación, pero el factor fundamental que termina decidiendo la utilización de PostgresSQL se basa en las funcionalidades disponibles en dicho motor de base de datos para soportar un sistema de las características

del sellado digital de tiempo.

8.3. Decisiones de implementación

En esta sección explicaré las decisiones de implementación relacionada con la infraestructura.

8.3.1. Integración con PKIGrid UNLP

PKIGrid UNLP[59] es la infraestructura que soporta las actividades de e-ciencia de la comunidad académica Argentina. La autoridad de certificación montada en la Universidad Nacional de La Plata para brindar servicio de firma digital no puede ser utilizada para obtener un certificado digital para instalar en el servidor de sellado de tiempo.

Figura 8.2: PKIGrid UNLP

En los comienzos de la implementación se planteó la integración de este nuevo servicio de sellado de tiempo con la infraestructura de e-ciencia que se encontraba disponible en la UNLP. A partir de ese momento, comencé a realizar pruebas de integración entre el servicio prototípico y el servicio que ya se encuentra en producción. Los análisis iniciales involucraron el estudio de la Política de Certificación, conocida como CP por su acrónimo en inglés. y la Declaración de las Prácticas de Certificación, conocida como CPS por el motivo explicado anteriormente.

Según dichos documentos, la CA de UNLP PKIGrid emite certificados para actividades de e-ciencia realizadas dentro de los parámetros de la UNLP Grid. Los tipos de certificados que esta CA emite son certificados personales, de servidor y de servicio.

Si analizamos el fin para el cual son emitidos los certificados, los certificados para una Autoridad de Sellado de Tiempo que preste soporte para los servicios de e-ciencia se enmarcan dentro de los prescrito por la Autoridad de Certificación de la UNLP.

Por la tanto, a nivel normativo, los certificados necesarios para el servicio implementado en este trabajo podrían ser emitidos por el servicio de firma digital existente.

A continuación, realicé un análisis de compatibilidad entre los certificados emitidos por la PKIGrid y las necesidades del producto OpenTSA, el cual resultó seleccionado para brindar el servicio detallado en la RFC 3161. Para ello se analizaron los diversos perfiles de certificados que actualmente está brindando PKIGrid UNLP.

Se concluyó que la restricción más importante se centra en los campos del certificado X.509 para el perfil de certificado destinado a los servicios, que sería el que se debería adecuar al que se necesita para la autoridad de sellado de tiempo. Las extensiones X.509 v3 que están presentes en el certificado para servidor o servicios es la siguiente:

Campo	Descripción
Basic Constraints:	critical, ca: false
Subject Key Identifier:	Hash
Authority Key Identifier:	Keyid
Subject Alternative Name:	DNS, Email
Key Usage:	critical, digitalSignature, KeyEncipherment, dataEncipherment
Extended Key Usage:	serverAuth, clientAuth, timeStamping
Netscape Cert Type:	SSL Server, SSL Client
Netscape Comment:	STRING
CRL Distribution Points:	URI (CRL)
Certificate Policies:	OID
Issuer alternative Name:	Email
nsRevocationUrl:	URI
NsCaPolicyUrl:	URI

Analizando la librería de OpenSSL, específicamente la verificación del certificado digital destinado a la autoridad de sellado de tiempo, se encuentra un requerimiento para la TSA relacionada con los valores de los campos Key Usage y Extended Key Usage. Para que un certificado pueda ser instalado, debe tener en los siguientes valores:

Campo	Descripción
Key Usage:	critical, digitalSignature
Extended Key Usage:	timeStamping

Este inconveniente, que no puede ser subsanado fácilmente debido a que debería modificarse toda la Política de Certificación de PKIGrid UNLP, conlleva a tener que instalar una nueva instancia de una autoridad de certificación para poder emitir certificados que cumplan los requisitos en las extensiones de X.509.

Para ello, utilicé el producto OpenCA, generando una nueva autoridad de certificación, configurada de manera tal que pueda emitir un certificado para el perfil de Servicios cumpliendo con los requisitos impuestos por OpenSSL.

Como alternativa a esta solución, es posible recompilar OpenSSL modificando las funciones que verifican los campos de extensión y de esa forma poder utilizar los certificados emitidos por PKIGrid UNLP, ya que los mismos tienen como uno de sus valores en el campo de extensión correspondiente el sellado de tiempo.

8.3.2. Elección del protocolo de transporte y mecanismo de sellado de tiempo

La norma ETSI TS 101 861 obliga a disponer de un protocolo en línea para la Autoridad de Sellado de Tiempo, por lo que la decisión de utilizar el protocolo de sellado

vía HTTP, a través del servicio implementado por OpenTSA, se adecúa perfectamente a las reglamentaciones. Es importante resaltar que el mecanismo de sellado de tiempo utilizado por este producto es un mecanismo de sellado simple, basado en una autoridad de certificación.

8.3.3. Sincronización de relojes

Uno de los componentes transcendentales en la arquitectura de una Autoridad de Sellado de Tiempo es la fuente confiable de tiempo a utilizar para sellar los requerimientos recibidos. Para ello, se va a utilizar el protocolo NTP para mantener sincronizada la hora de los servidores involucrados en la arquitectura de la Autoridad de Sellado de Tiempo, entre ellos, el equipo que se encargará de emitir los sellos. Como fuente principal de tiempo se va a utilizar el servidor central que brinda la hora oficial para la Universidad Nacional de La Plata. El mismo se encuentra ubicado en el núcleo de la red y obtiene la hora de referencia del reloj del Instituto Argentino de Radioastronomía (IAR), el cual es un reloj Stratum 0 de tipo GPS. Además, se tiene definido mecanismos de redundancia de relojes de referencia vía Internet e Internet 2 con el fin de afrontar posibles cortes de disponibilidad que pueda haber con la red del IAR. De esa manera, ante un inconveniente en el reloj GPS, se puede obtener la hora mediante caminos alternativos a través de la red.

La figura 8.3 muestra la arquitectura de relojes establecida para la sincronización.

8.3.4. Algoritmos de cifrado

Hashing

Las restricciones impuestas por la norma de la ETSI mencionan tres protocolos de hashing o resumen como utilizables en el servicio de sellado de tiempo: SHA-1, MD5 o RIPEMD-160, pero recomienda descartar MD5 debido a los problemas de seguridad que fueron descubiertos en los últimos años, en relación a las colisiones encontradas.

Entonces, las alternativas existentes se reducen a SHA-1 o RIPEMD-160. Este último protocolo es un desarrollo surgido en los ámbitos académicos, a diferencia de la familia de protocolos SHA, creados por la Agencia de Seguridad Nacional de Estados Unidos. El producto que va a ser utilizado para implementar el servicio de sellado de tiempo, OpenTSA, tiene soporte para ambos protocolos, por lo tanto no impone una restricción al respecto.

La aparición de la RFC 5816, tal como fue mencionado anteriormente, produce modificaciones en los algoritmos de hashing o resumen que se permiten utilizar en algunos campos del sello de tiempo. La nueva norma impone que debe utilizarse SHA-1, por lo que esta discusión relacionada con los diferentes algoritmos que debería soportar esta solución se termina con esta nueva directiva.

Por lo tanto, el único algoritmo de hashing disponible será SHA-1.

Firma

La ETSI TS 101 861 indica que el algoritmo de firma que se debe soportar es SHA-1 con RSA. La longitud de las claves debe ser como mínimo 1024 bits, pero aclara que

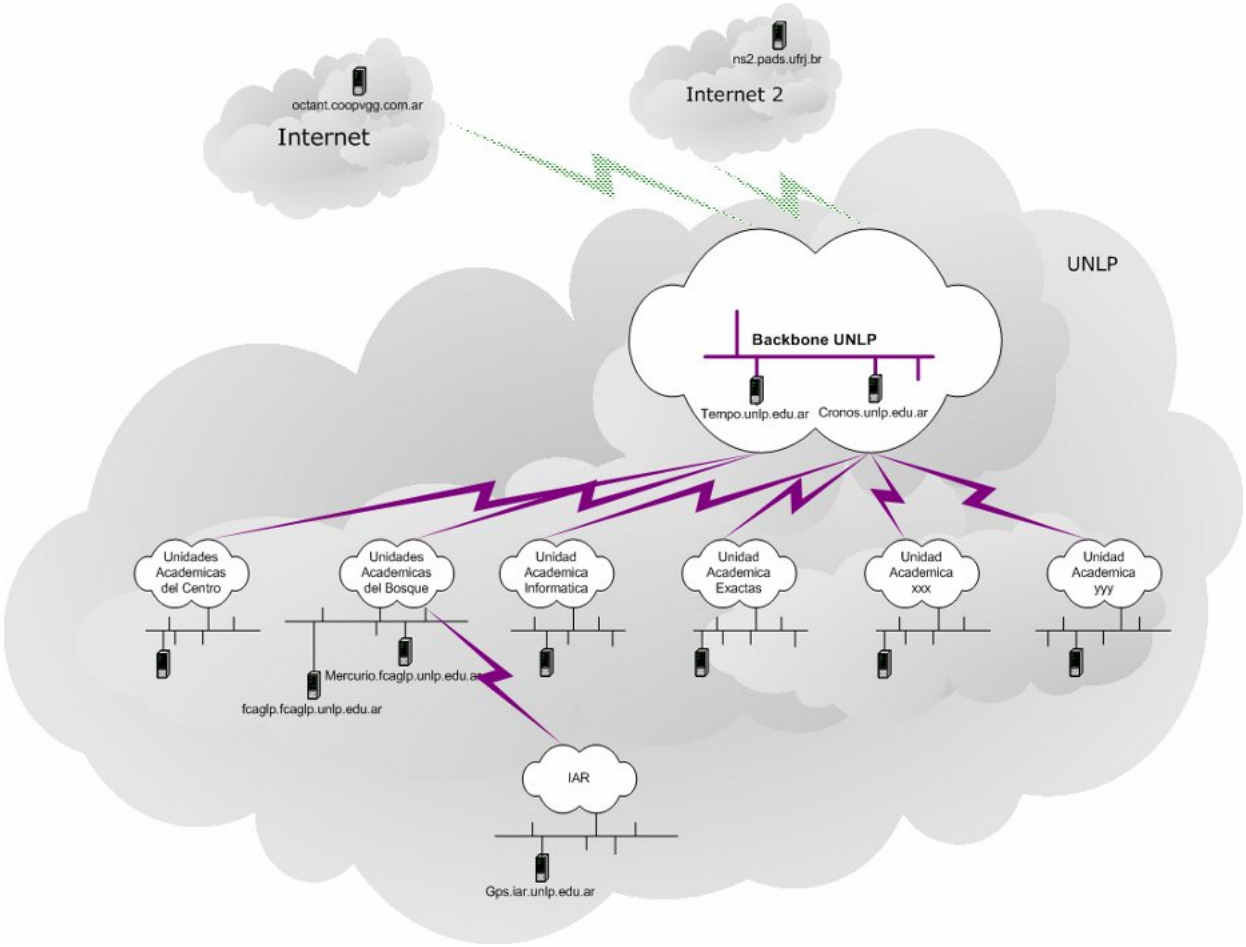

Figura 8.3: Arquitectura de sincronización de relojes

es deseable soportar claves de 2048 bits. Ante los recientes acontecimientos que hacen mención a inconvenientes en el cifrado de 1024 bits[60], decidí utilizar claves de 2048 bits de longitud. La autoridad de certificación puesta en marcha para este trabajo, genera certificados cuyas claves cumplen con este requisito.

8.3.5. Precisión de los relojes

Las distintas especificaciones existentes no imponen restricciones sobre el desvío mínimo que puede tener la fuente de tiempo. Se va a establecer como parámetro un desvío máximo de 500 milisegundos. En caso de que sea imposible la obtención de la exactitud requerida por parte de la fuente de tiempo a través de los distintos caminos establecidos, el token de sello de tiempo no puede ser emitido. El valor de 500 ms se ha seleccionado luego de haber analizado las distintas especificaciones de la autoridades de sellado de tiempo en funcionamiento, cuya mayoría utilizaba este valor como el máximo desvío admitido.

8.4. RFC 3161

Las solicitudes y respuestas de sellos se se adhieren a la sintaxis de la especificación "RFC3161 Time Stamp Protocol (TSP)" descripto en el Apartado 3.4. "Time-Stamp Protocol via http" de la especificación, con las restricciones impuestas por la norma ETSI TS 101 861.

Capítulo 9

Implementación

El presente capítulo contiene el desarrollo de todas tareas de implementación realizadas. Entre las configuraciones realizadas se encuentran la instalación y configuración de los servidores de base de datos, de los servidores web, del servicio de NTP y de la Autoridad de Certificación para la emisión de certificados. Además se muestran los principales lineamientos de la aplicación web desarrollada para permitir el acceso al servicio por parte de terceros. El capítulo finaliza con diversas pruebas de funcionamiento para verificar la funcionalidad de la aplicación y en particular el cumplimiento de las restricciones resultantes del análisis realizado en el capítulo anterior.

La figura 9.1 muestra la arquitectura definida para el servicio de sellado de tiempo.

9.1. Tareas de implementación realizadas

A continuación se listan las tareas de implementación y configuración realizadas.

- Instalación de una Autoridad de Certificación utilizando OpenCA como herramienta.
- Instalación de PostgresSQL y el sistema de replicación pgpool-II.
- Instalación y optimización de NTP.
- Compilación de OpenSSL con el modulo de Sellado de Tiempo.
- Integración de Apache con OpenTSA.
- Desarrollo de un Frontend Web.

9.1.1. Instalación de una Autoridad de Certificación utilizando OpenCA

El proyecto OpenCA es un producto de código abierto que provee una infraestructura para poner en marcha una Autoridad de Certificación para la emisión de Certificados Digitales.

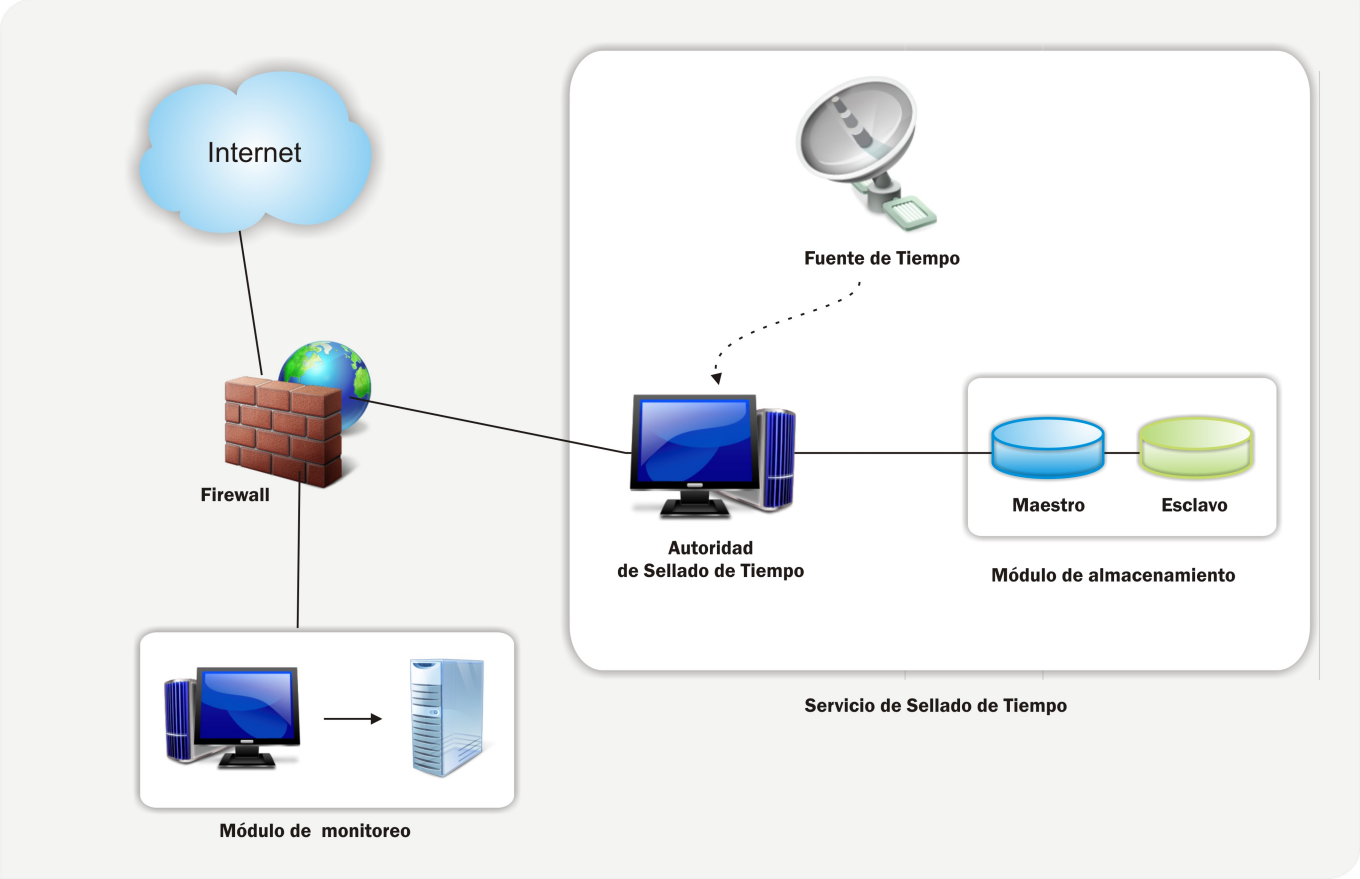

Figura 9.1: Arquitectura del servicio de sellado de tiempo

La PKIGrid creada en la Universidad Nacional La Plata, para dar soporte a las actividades de e-ciencia de la comunidad académica, utiliza este producto como plataforma tecnológica principal para brindar el servicio.

Ante la necesidad de emitir certificados digitales para los diversos componentes de la Autoridad de Sellado de Tiempo, he decidido instalar el producto OpenCA, en su versión 1.0.2.

Una vez realizadas las modificaciones en las plantillas que representan los distintos tipos de certificados que se pueden emitir, he inicializado la Autoridad de Certificación, he generado el certificado raíz y luego he emitido los certificados correspondientes al frontend de consulta y el certificado de la TSA, que es el que se va a utilizar para firmar todos los requerimientos de sellado que se soliciten.

En esta etapa es donde he aplicado el análisis previo de las extensiones X.509, modificando los archivos de extensiones de OpenCA para cumplir con los requerimientos de OpenTSA para poder desplegar y utilizar el servicio.

En particular, he modificado el tipo de certificado correspondiente al rol “Service”, para que los campos de extensión sean los adecuados para el certificado de una Autoridad de Sellado de Tiempo.

Figura 9.2: OpenCA

9.1.2. Instalación de PostgreSQL

Para poder almacenar los sellos de tiempo emitidos, es necesario contar con una base de datos relacional, y he elegido PostgreSQL como motor de base de datos, principalmente por el soporte de replicación y alta disponibilidad que este producto brinda, como he explicado anteriormente.

En relación a esto, he instalado dos instancias del servidores de PostgreSQL versión 8.3.9 con soporte de replicación. A su vez, he instalado y configurado el producto Pgpool-II, para poder soportar replicación y alta disponibilidad. Mediante este producto, cuyo funcionamiento principal, se observa en la figura 9.4, es posible contar con un mecanismo de replicación y tolerancia a fallos que forma parte de los requisitos de un servicio que debe estar disponible 7x24.

La figura 9.4 ilustra la arquitectura implementada para el módulo de almacenamiento. En la misma se observa la existencia de un servicio que actúa como interfaz entre las bases de datos que conforman el modelo de replicación y las aplicaciones que hacen uso del mismo; las dos instancias que se encuentran replicadas en forma continua no son accesibles, solamente se publica la interfaz de base de datos que funciona como frontend. Cuando se realiza una operación sobre la base de datos, el servicio provisto por Pgpool-II se encarga de interactuar con las dos instancias que administra sin que el usuario final del servicio de base de datos detecte la presencia de esta solución.

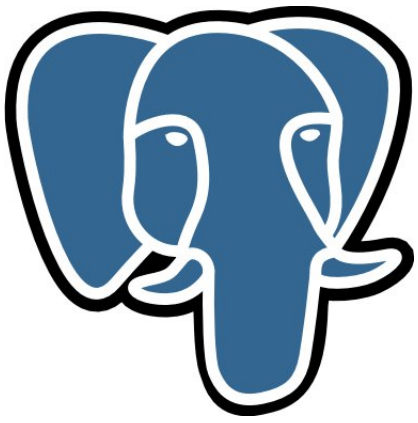

Figura 9.3: Servidor de Base de datos PostgreSQL

9.1.3. Configuración de NTP

Para implementar el servidor de NTP, he utilizado el paquete proporcionado por Debian, desarrollado por el equipo de desarrollo del sistema operativo. Este producto,

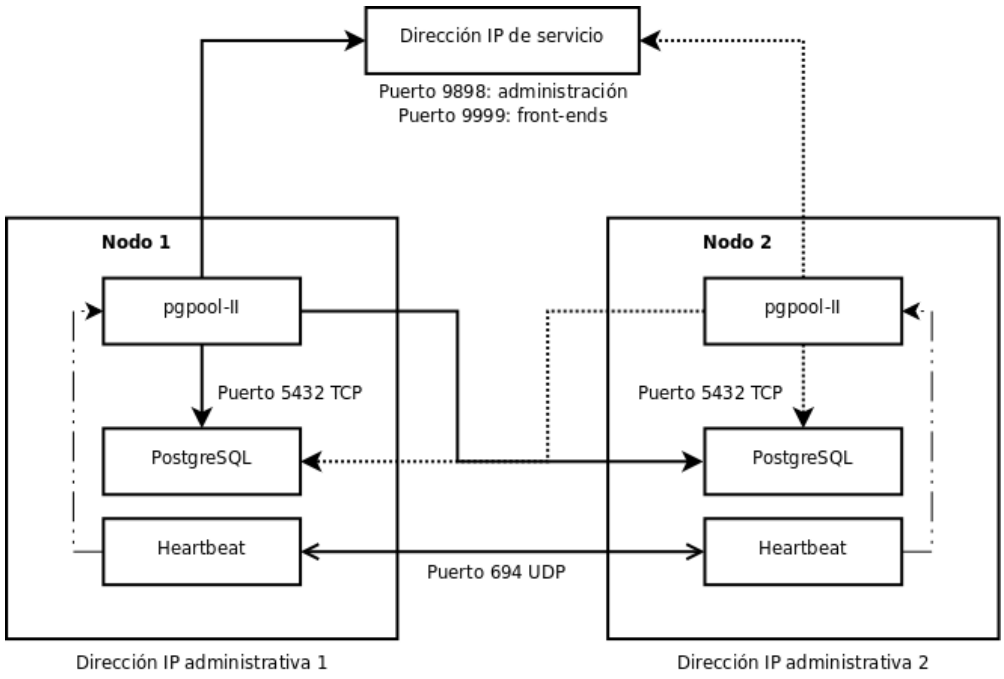

Figura 9.4: Arquitectura redudante de almacenamiento

cuya versión instalada es la 4.2.4, permite la sincronización via Internet o una red local, además de interpretar diversas señales de tiempo entre las que se encuentra GPS.

A continuación destaco algunos parámetros de configuración del producto:

```
#Drift file
driftfile /var/lib/ntp/ntp.drift

#Estadisticas diarias de peers, loop y clock....
statsdir /var/log/ntpstats/

statistics loopstats peerstats clockstats
filegen loopstats file loopstats type day enable
filegen peerstats file peerstats type day enable
filegen clockstats file clockstats type day enable

#Servidores
#Internet
server pool.ntp.org iburst dynamic
#Internet 2
server chronos1.umt.edu iburst dynamic
#Locales
server cronos.unlp.edu.ar iburst dynamic
server gps.iar.unlp.edu.ar iburst dynamic


#Restricciones de acceso
restrict -4 default kod notrap nomodify nopeer noquery
restrict -6 default kod notrap nomodify nopeer noquery
restrict 127.0.0.1
```

En esta configuración, se proporcionan las directivas para la configuración de las estadísticas del servicio, que van generarse diariamente. Por otro lado, se listan los servidores con los cuales se va a sincronizar la hora de este servidor de tiempo. Para ello

se utilizan diversas fuentes, tanto pertenecientes a la infraestructura de la UNLP, como relojes disponibles en Internet e Internet 2. De esta forma, se proporciona un mecanismo de sincronización de alta disponibilidad y tolerancia a fallos de conectividad.

A continuación, se muestra el resultado de la consulta de los peers utilizados por el servidor NTP configurado; en la misma se pueden observar las direcciones IP de los servidores, el retraso y el desplazamiento de UTC, entre otros parámetros.

```
 ntpq> peers
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
+mail.wholenet.c 91.189.94.4      3 u  337 1024  377  173.759  -96.711   0.883
+br1.umtnet.umt. 204.34.198.41    2 u  393 1024  377  279.824   13.367   1.170
 server.cespi.un 163.10.43.42     2 u  294 1024  377    0.568    3.982   0.919
*gps.iar.unlp.ed .GPS.            1 u  538 1024  377    2.445    3.685   7.127
ntpq>
```

En el listado de servidores, encontramos dos servidores pertenecientes a la UNLP, un servidor perteneciente al proyecto ntp.org, el cuál es alcanzado por Internet comercial y un servidor perteneciente a una casa de estudios de Estados Unidos, que es alcanzado por la red de Internet 2, mediante un enlace con la Red de Interconexión Universitaria (RIU).

9.1.4. Compilación de OpenSSL

Al momento de comenzar con las tareas de implementación, el soporte de timestamping no era soportado en forma nativa por las versiones de OpenSSL existentes; era necesario contar con un patch brindado por OpenTSA para contar con dichas características en OpenSSL.

A partir de la versión 1.0.0, que fue liberada el 29 de marzo de este año, el soporte para timestamping ya viene incorporado, por lo que no es necesaria la modificación de OpenSSL.

Debido a que Debian Lenny, la versión del sistema operativo utilizada, no contiene en sus repositorios la última versión liberada, fue necesario compilar e instalar la versión 1.0.0a.

Figura 9.5: Publicidad de OpenSSL resaltando la característica de Open Source

9.1.5. Integración de OpenTSA con Apache

Para poder hacer uso de OpenTSA, es necesario integrarlo a Apache, de forma tal que las peticiones a la TSA lleguen mediante el protocolo HTTP. Para ello, fue necesario compilar OpenTSA, indicándole la versión de OpenSSL a utilizar y los diversos motores de base de datos que debía soportar, entre ellos PostgreSQL, que resultó seleccionado para el prototipo.

Además, fue necesario compilar Apache en su versión 2.2.9, para brindar soporte de SSL y de librerías compartidas, con el objetivo de compilar apropiadamente el producto OpenTSA. Se realizaron pruebas de compatibilidad con la versión provista por el sistema operativo Debian, pero los resultados fueron negativos; por eso opté por la compilación del mismo para integrarlo con OpenTSA.

A continuación se presenta el archivo de configuración de la Autoridad de Sellado de Tiempo:

```
#Archivo de almacenamiento de los numeros de serie. Es actualizado cada vez que se
    generada una respuesta.
TSASerialFile conf/tsaserial

#Certificado utilizado para firmar los sellos de tiempo
TSACertificate /root/server.pem

#Certificado raiz
TSACertificateChain /root/cacert.pem

#Clave privada de la TSA
TSAKey /root/serverPrivada.pem

#Passphrase de la clave privada de la TSA
TSAKeyPassPhrase On

#OID de la ípoltica por defecto de la TSA. Este únmero es un únmero al azar. No fue
    definido un OID para este trabajo
TSADefaultPolicy 1.1.2

#Otras ípolticas aceptadas en caso que el cliente requiera un sello con esta ípoltica.
    Este únmero es un únmero al azar. No fue definido un OID para este trabajo.
TSAPolicies 1.1.3 1.1.4

#Lista de algoritmos de sellado soportados
TSAMessageDigests sha1

#óPrecisin del sellado de tiempo en milisegundos
TSAAccuracy 0 500 0

#óPrecisin de digitos de segundos
TSAClockPrecisionDigits 0

#Campo de orden
TSAOrdering Off

# Incluir nombre de la TSA en las respuestas
TSAIncludeName On

#Modulo de base de datos
TSADBModule PostgreSQL

#Servidor de base de datos
TSAPostgreSQLHost localhost

#Puerto de Mysql
TSAPostgreSQLPort 9999

#Usuario de la base de datos
```

```
#TSAPostgreSQLUser root

#Nombre de la base de datos
#TSAPostgreSQLDatabase tsa

#Clave del motor de base de datos
TSAPostgreSQLPassPhrase On

</IfModule>
```

Figura 9.6: Servidor Web Apache

9.1.6. Frontend Web

Un componente fundamental en el desarrollo de este prototipo de Autoridad de Sellado de Tiempo, es la creación de una interfaz web que permita la interacción entre el usuario y la aplicación de manera simple. Para ello, he diseñado una aplicación Web sencilla que puede ser utilizada por dos tipos de perfiles de usuario; para ello se cuenta con una interfaz pública y una interfaz privada, con diversas funcionalidades que se van a explicar a continuación. En primer término, es preciso definir las dos interfaces con los que cuenta el sitio web desarrollado.

- Interfaz pública: Esta interfaz involucra a los usuarios que quieren interactuar con la Autoridad de Sellado de Tiempo con el objetivo de consultar la validez de un sello de tiempo, visualizar los sellos emitidos por la Autoridad o descargar los certificados digitales correspondientes a la TSA y a la CA que ha emitido los certificados.

- interfaz de monitoreo: Esta interfaz permite utilizar las funciones de monitoreo de los servicios involucrados en el prototipo; el funcionamiento de cada uno de estos sistemas de monitoreo serán tratados en el capítulo siguiente.

La funcionalidad de Administración es restringida haciendo uso del módulo de autenticación proporcionado por Apache; este tema será explicado en el próximo capítulo.

Para el desarrollo de esta interfaz Web he utilizado diversas tecnologías actuales entre las que se puede mencionar a Perl como lenguaje de scripting.

La interfaz pública del frontend Web cuenta con varias opciones que comprenden las funcionalidades básicas que un servicio de Sellado de Tiempo debería brindar a un usuario particular.

- **Listar tokens emitidos**: Mediante esta opción es posible listar los tokens emitidos, visualizando algunos campos importantes que comprenden el sello de tiempo; en la tabla se visualiza el número de serie del token emitido, la fecha en formato UTC que indica el momento en el cual fue sellado el dato, el hash del dato enviado para su sellado y finalmente el algoritmo de hashing o resumen utilizado para realizar el hash anteriormente visualizado.

- **Validar tokens**: Con esta función un usuario que posea un sello de tiempo emitido por esta Autoridad de Sellado de Tiempo, puede verificar la validez del sello. Para ello, el usuario debe subir al servidor el sello de tiempo, en un archivo con extension **.tsr** y aguardar por la respuesta del servidor.
- **Obtener certificados**: Todas las Entidades Certificadoras entre las que podemos enmarcar a las Autoridades de Sellado de Tiempo, deben ofrecer para su descarga los certificados digitales utilizados para la firma de los sellos de tiempo además de permitir la descarga del certificado raíz con el cual el certificado de firma fue emitido. Esta funcionalidad es proporcionada desde la opción "Certificados".

Para la creación de este interfaz web he utilizado diversas herramientas y tecnologías para facilitar el desarrollo actual y una posterior ampliación de la funcionalidad del mismo: Perl, Ajax, PHP, CSS, entre otras. Más específicamente, he utilizado Apache 2.2.9, PHP 5.2.6, JQuery 1.2.6 y Perl 5.10. Con respecto a Apache, para esta parte de la implementación, he utilizado la versión precompilada disponible en los repositorios de Debian.

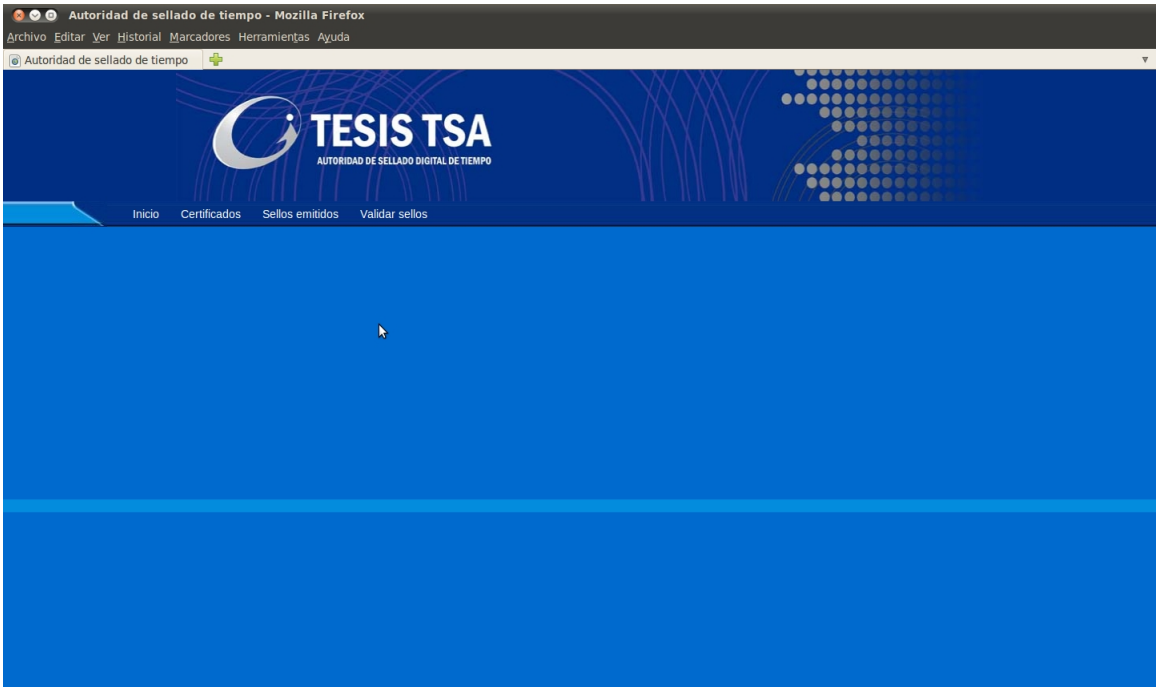

Figura 9.7: Interfaz web de la TSA

9.1.7. Perfil de la TSA

El perfil del certificado de la TSA, utilizado en la firma de los token de tiempo, se ajusta a lo especificado por el IETF en RFC 3161. En el cuadro 9.1 que se encuentra a continuación, se detallan los campos básicos de este perfil.

El servicio de Sellado de Tiempo es accesible vía HTTP y HTTPS a través de las siguientes URL **http://tsa.linti.unlp.edu.ar/tsa** y **https://tsa.linti.unlp.edu.ar/tsa**, respectivamente.

Nombre del campo	Valor	
Version	Version 3	
Serial Number	Valor único para todos los certificados emitidos	
Signature Algorithm	sha1withRSAEncryption (1.2.840.113549.1.1.5)	
Issuer	Common Name(CN)	Tesis Alejandro Sabolansky
	Organizacional Unit Name	UNLP
	Organization Name	LINTI
	Country	AR
Not before	Fecha de inicio del periodo de validez del certificado	15 de agosto de 2010 00:15:16 GMT
Not After	Fecha de fin del periodo de validez del certificado	14 de septiembre de 2011 00:15:16 GMT
Subject	Common Name (CN)	tsa.linti.unlp.edu.ar
	Organizacional Unit Name	UNLP
	Organization Name	UNLP
	Country	AR
Subject Public Key Info	Codificado de acuerdo al RFC 2459, contiene información de la clave publica RSA. Tamaño 512 bits	
Signature	Certificado de firma. Generado y codificado acorde al RFC 2459	
Uso de la clave	Firma digital (80) Marcado como crítico	
Uso extendido de la clave	Impresión de fecha (1.3.6.1.5.5.7.3.8) Marcado como crítico	

Cuadro 9.1: Perfil del certificado de la TSA

9.1.8. Certificados emitidos

A continuación se muestra el contenido del certificado digital de la TSA, del servidor web de la TSA y de la entidad raíz que firmó los dos certificados anteriores.

Certificado digital de la TSA

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 28 (0x1c)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: emailAddress=soporte@linti.unlp.edu.ar,CN=Tesis Alejandro Sabolansky,
            OU=LINTI,O=UNLP,ST=Buenos Aires,L=La Plata,C=AR
        Validity
            Not Before: Aug 15 00:15:16 2010 GMT
            Not After : Sep 14 00:15:16 2011 GMT
        Subject: CN=tsa.linti.unlp.edu.ar,L=La Plata,OU=UNLP,C=AR
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:b7:a6:f6:38:3e:02:0e:49:55:22:eb:c1:cc:9f:
                    66:d4:b6:41:a1:d6:40:45:52:79:d5:12:eb:f9:84:
                    38:61:22:4b:3d:31:12:90:d5:ac:e6:e9:52:30:ce:
                    c2:9e:95:a4:ae:56:63:f0:43:f7:a6:d9:94:c9:95:
                    d7:55:f2:d5:b9:2c:02:11:5d:de:fb:bc:b4:b2:5c:
                    ac:21:ae:cc:f3:12:b2:c1:f2:25:46:a8:5d:56:96:
                    71:d6:d5:33:51:d9:3c:44:af:31:fd:01:60:04:c5:
                    3f:ed:2c:93:a6:6f:38:24:ea:29:e8:f4:b2:d5:77:
                    3d:d6:4e:2f:e5:9b:72:46:16:84:08:78:7d:0b:97:
                    81:e7:ad:1d:96:76:7c:27:0c:11:39:78:e2:34:bd:
                    fa:5d:49:31:1d:0a:ac:19:71:db:f0:7f:aa:c6:d3:
                    81:eb:c2:49:20:6f:9b:bf:55:d3:e4:2c:26:8c:0f:
                    7e:5f:09:89:0d:31:ce:5d:ae:2c:f9:69:2f:49:40:
                    4e:c6:2d:e3:f8:74:c4:0d:ce:47:f5:29:df:29:22:
                    02:be:e8:df:73:22:e4:58:6e:c8:f0:d4:9c:52:bb:
                    d8:6a:73:ca:20:74:2b:41:21:38:07:b9:af:dd:4c:
                    9b:87:1d:30:1e:1c:7d:4f:f9:a7:15:30:0c:74:52:
                    c9:8f
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints: critical
```

```
                CA:FALSE
            Netscape Cert Type:
                SSL Client, SSL Server
            X509v3 Key Usage: critical
                Digital Signature
            X509v3 Extended Key Usage: critical
                Time Stamping
            Netscape Comment:
                UNLP Certification Authority Service Signed
            X509v3 Subject Key Identifier:
                BC:A5:D5:F2:7B:3E:3B:51:66:52:65:F5:49:44:05:76:06:23:E6:7F
            X509v3 Authority Key Identifier:
                keyid:03:6D:5E:D7:8B:E5:44:DC:22:CA:32:6B:16:F0:69:4F:AA:F6:2C:ED

            X509v3 Subject Alternative Name:
                email:soporte@linti.unlp.edu.ar, DNS:tsa.linti.unlp.edu.ar
            X509v3 Issuer Alternative Name:
                email:camanager2@pkigrid.unlp.edu.ar
            Netscape Revocation Url:
                @NS_RevocationUrl@
            Netscape CA Policy Url:
                @NS_CAPolicyUrl@
            X509v3 CRL Distribution Points:
                URI:http://ca.tesisalejandro.linti.unlp.edu.ar/pki/pub/crl/cacrl.crl

            X509v3 Certificate Policies:
                Policy: 1.2.840.113612.5.4.2.3.1.0.2.7

    Signature Algorithm: sha1WithRSAEncryption
        56:6d:b9:9e:70:0b:c4:8c:66:9b:6c:d0:0f:12:a9:75:29:89:
        33:5a:87:f6:3c:de:4e:4f:d2:e0:7a:60:94:49:d9:a5:14:48:
        ef:58:f5:31:8e:fd:37:d2:ad:90:85:2c:da:40:82:94:66:df:
        3d:17:d9:7e:dd:f0:4a:ce:5d:bf:81:2f:99:85:bc:76:08:7b:
        0d:41:63:a6:5b:06:6a:d2:d1:45:c4:47:28:82:98:59:88:65:
        5c:b5:46:8b:fa:75:c3:5d:69:54:6a:1b:60:3c:df:79:15:f8:
        c6:93:6a:02:01:2e:82:78:90:0d:69:95:74:94:f7:89:f3:19:
        00:f9:6a:a3:15:86:d0:e4:d9:ef:a6:5f:e6:47:59:1e:88:9a:
        38:01:2b:b7:51:0e:34:59:29:30:93:6c:61:a0:8f:db:83:13:
        85:5c:0e:4b:49:1c:68:e1:0c:ff:fa:c0:8a:a7:81:50:b1:b7:
        d6:7f:00:41:87:5f:71:a9:2c:db:f1:c6:b9:86:fa:ad:39:df:
        7f:92:0c:9c:48:df:7e:19:b5:55:df:f1:f2:dd:ea:84:ad:5d:
        84:53:3b:7d:13:88:7d:d7:c2:74:73:05:dc:f5:6f:a7:5b:f7:
        3d:56:b9:07:77:5b:69:b7:8c:53:24:36:b4:51:f7:6c:01:a8:
        a3:d4:ea:57
-----BEGIN CERTIFICATE-----
MIIFVzCCBD+gAwIBAgIBHDANBgkqhkiG9w0BAQUFADCBpTELMAkGA1UEBhMCQVIx
ETAPBgNVBAcMCExhIFBsYXRhMRUwEwYDVQQIDAxCdWVub3MgQWlyZXMxDTALBgNV
BAoMBFVOTFAxDjAMBgNVBAsMBUxJTlRJMSMwIQYDVQQDDBpUZXNpcyBBbGVqYW5k
cm8gU2Fib2xhbnNreTEoMCYGCSqGSIb3DQEJARYZc29wb3J0ZUBsaW50aS51bmxw
LmVkdS5hcjAeFw0xMDA4MTUwMDE1MTZaFw0xMTA5MTQwMDE1MTZaME8xCzAJBgNV
BAYTAkFSMQ0wCwYDVQQLDARVTkxQMREwDwYDVQQHDAhMYSBQbGF0YTEeMBwGA1UE
AwwVdHNhLmxpbnRpLnVubHAuZWR1LmFyMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8A
MIIBCgKCAQEAt6b2OD4CDklVIuvBzJ9m1LZBodZARVJ51RLr+YQ4YSJLPTESkNWs
5ulSMM7CnpWkrlZj8EP3ptmUyZXXVfLVuSwCEV3e+7y0slysIa7M8xKywfIlRqhd
VpZx1tUzUdk8RK8x/QFgBMU/7SyTpm84JOop6PSy1Xc91k4v5ZtyRhaECHh9C5eB
560dlnZ8JwwROXjiNL36XUkxHQqsGXHb8H+qxtOB68JJIG+bv1XT5CwmjA9+XwmJ
DTHOXa4s+WkvSUBOxi3j+HTEDc5H9SnfKSICvujfcyLkWG7I8NScUrvYanPKIHQr
QSE4B7mv3Uybhx0wHhx9T/mnFTAMdFLJjwIDAQABo4IB5TCCAeEwDAYDVR0TAQH/
BAIwADARBglghkgBhvhCAQEEBAMCBsAwDgYDVR0PAQH/BAQDAgeAMBYGA1UdJQEB
/wQMMAoGCCsGAQUFBwMIMDoGCWCGSAGG+EIBDQQtFitVTkxQIENlcnRpZmljYXRp
b24gQXV0aG9yaXR5IFNlcnZpY2UgU2lnbmVkMB0GA1UdDgQWBBS8pdXyez47UWZS
ZfVJRAV2BiPmfzAfBgNVHSMEGDAWgBQDbV7Xi+VE3CLKMmsW8GlPqvYs7TA7BgNV
HREENDAygRlzb3BvcnRlQGxpbnRpLnVubHAuZWR1LmFyghV0c2EubGludGkudW5s
cC5lZHUuYXIwKQYDVR0SBCIwIIEeY2FtYW5hZ2VyMkBwa2lncmlkLnVubHAuZWR1
LmFyMCEGCWCGSAGG+EIBAwQUFhJATlNfUmV2b2NhdGlvblVybEAwHwYJYIZIAYb4
QgEIBBIWEEBOU19DQVBvbGljeVVybEAwUQYDVR0fBEowSDBGoESgQoZAaHR0cDov
L2NhLnRlc2lzYWxlamFuZHJvLmxpbnRpLnVubHAuZWR1LmFyL3BraS9wdWIvY3Js
L2NhY3JsLmNybDAbBgNVHSAEFDASMBAGDiqGSIb3TAUEAgMBAAIHMA0GCSqGSIb3
DQEBBQUAA4IBAQBWbbmecAvEjGabbNAPEql1KYkzWof2PN5OT9LgemCUSdmlFEjv
WPUxjv030q2QhSzaQIKUZt89F9l+3fBKzl2/gS+Zhbx2CHsNQWOmWwZq0tFFxEco
gphZiGVctUaL+nXDXWlUahtgPN95FfjGk2oCAS6CeJANaZV0lPeJ8xkA+WqjFYbQ
```

```
5Nnvpl/mR1keiJo4ASu3UQ40WSkwk2xhoI/bgxOFXA5LSRxo4Qz/+sCKp4FQsbfW
fwBBh19xqSzb8ca5hvqtOd9/kgycSN9+GbVV3/Hy3eqErV2EUzt9E4h918J0cwXc
9W+nW/c9VrkHd1tpt4xTJDa0UfdsAaij1OpX
-----END CERTIFICATE-----
```

Certificado raíz

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            a5:19:c3:59:1d:b6:60:cd
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: emailAddress=soporte@linti.unlp.edu.ar,CN=Proyecto ANMAT,OU=LINTI,O=
            UNLP,ST=Buenos Aires,L=La Plata,C=AR
        Validity
            Not Before: May  9 22:44:00 2010 GMT
            Not After : May  4 22:44:00 2030 GMT
        Subject: emailAddress=soporte@linti.unlp.edu.ar,CN=Proyecto ANMAT,OU=LINTI,O=
            UNLP,ST=Buenos Aires,L=La Plata,C=AR
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:e3:d2:f4:a6:0f:4a:ee:31:f5:c6:fa:cc:d1:24:
                    27:a7:5a:8b:9d:e3:d6:c0:9a:b8:5a:b3:d2:d5:c8:
                    e9:9b:e3:fc:86:23:ea:22:0d:d9:ef:ca:27:19:78:
                    cb:8a:db:5c:f7:07:c6:9b:3e:bd:33:50:ef:89:32:
                    86:a0:11:bc:5f:43:4c:fe:9b:f3:4f:01:4b:55:7d:
                    5f:78:34:38:d9:f4:c0:32:bd:57:42:9e:b0:e5:a9:
                    ab:1d:4c:61:de:ee:ec:5a:13:c0:88:a4:54:af:45:
                    26:42:f3:75:99:a0:32:75:9d:72:b5:f0:77:50:33:
                    9e:e8:c1:36:d1:57:c9:c9:93:6d:fa:7f:20:de:d3:
                    df:02:35:c7:34:13:e8:cd:85:17:ea:0b:3e:44:27:
                    8b:38:e5:5b:7b:a7:c4:e7:17:3c:e5:d2:12:ca:71:
                    16:ee:df:8d:c4:b6:49:59:21:b6:0d:27:d5:03:f6:
                    82:e3:e5:44:7d:e4:e3:ae:47:d3:ad:4f:66:d1:49:
                    e0:01:4e:06:22:60:b2:69:ec:ca:18:91:28:a5:e5:
                    a2:55:99:24:2c:e2:b3:e8:c9:5f:1b:13:51:03:af:
                    51:5d:ff:51:55:15:94:dc:90:7c:b4:27:ef:e5:c2:
                    f1:ae:8e:5c:8b:ba:4a:81:40:04:63:1c:01:e5:b3:
                    a1:11
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Subject Key Identifier:
                9B:F2:21:98:21:66:4D:24:9F:BC:E5:D1:9F:64:35:47:72:A9:5C:5C
            X509v3 Authority Key Identifier:
                keyid:9B:F2:21:98:21:66:4D:24:9F:BC:E5:D1:9F:64:35:47:72:A9:5C:5C

            X509v3 Key Usage: critical
    Certificate Sign, CRL Sign
            X509v3 Subject Alternative Name:
                email:root@localhost
            X509v3 CRL Distribution Points:
                URI:http://ca.tesisalejandro.linti.unlp.edu.ar/pki/pub/crl/cacrl.crl

    Signature Algorithm: sha256WithRSAEncryption
        4b:e4:a3:9e:6e:a9:dd:98:86:de:20:cc:ee:b7:0c:2b:32:28:
        eb:e7:9d:8b:0e:a9:37:23:6c:ae:42:29:91:ee:8a:b2:a4:13:
        b1:13:2b:ad:07:18:1a:c4:a7:61:2b:1c:35:42:9c:f4:0c:96:
        49:6b:48:0b:66:5d:89:53:8e:bc:2d:9a:fc:d8:7b:5c:7c:e2:
        3a:18:30:07:d7:7a:b1:88:3e:a1:ea:c3:6b:9a:68:67:04:4e:
        53:31:b3:70:e9:1f:15:d1:93:f0:6a:50:51:fd:1f:3f:8a:6d:
        89:d7:1d:0d:40:23:51:70:ae:8d:b9:61:65:ae:e9:39:72:a1:
        28:30:6e:2c:25:fb:a7:ab:ae:44:52:3b:52:3b:3d:bf:d0:3e:
```

```
        76:60:b9:73:c2:37:04:4d:70:ed:18:75:2b:40:5b:38:42:3d:
        11:4d:91:20:af:ad:a2:fa:96:a7:61:f9:18:79:b1:d3:af:34:
        15:9c:e9:1d:b3:37:f7:40:2c:4e:21:32:bd:12:62:5a:53:14:
        99:c4:79:5a:63:be:af:9f:35:59:02:b8:66:2c:84:ea:45:46:
        1d:56:3d:5a:be:5e:b2:b6:ce:ed:24:11:b3:f6:50:4a:eb:57:
        c0:47:da:b8:7a:f4:6f:f2:68:14:15:28:16:ed:f9:28:5f:a6:
        a9:f0:57:4e
-----BEGIN CERTIFICATE-----
MIIEcDCCA1igAwIBAgIJAKUZw1kdtmDNMA0GCSqGSIb3DQEBCwUAMIGZMQswCQYD
VQQGEwJBUjERMA8GA1UEBwwITGEgUGxhdGExFTATBgNVBAgMDEJ1ZW5vcyBBaXJl
czENMAsGA1UECgwEVU5MUDEOMAwGA1UECwwFTElOVEkxFzAVBgNVBAMMDlByb3ll
Y3RvIEFOTUFUMSgwJgYJKoZIhvcNAQkBFhlzb3BvcnRlQGxpbnRpLnVubHAuZWR1
LmFyMB4XDTEwMDUwOTIyNDQwMFoXDTMwMDUwNDIyNDQwMFowgZkxCzAJBgNVBAYT
AkFSMREwDwYDVQQHDAhMYSBQbGF0YTEVMBMGA1UECAwMQnVlbm9zIEFpcmVzMQ0w
CwYDVQQKDARVTkxQMQ4wDAYDVQQLDAVMSU5USTEXMBUGA1UEAwwOUHJveWVjdG8g
QU5NQVQxKDAmBgkqhkiG9w0BCQEWGXNvcG9ydGVAbGludGkudW5scC5lZHUuYXIw
ggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDj0vSmD0ruMfXG+szRJCen
Woud49bAmrhas9LVyOmb4/yGI+oiDdnvyicZeMuK21z3B8abPr0zUO+JMoagEbxf
Q0z+m/NPAUtVfV94NDjZ9MAyvVdCnrDlqasdTGHe7uxaE8CIpFSvRSZC83WZoDJ1
nXK18HdQM57owTbRV8nJk236fyDe098CNcc0E+jNhRfqCz5EJ4s45Vt7p8TnFzzl
0hLKcRbu343EtklZIbYNJ9UD9oLj5UR95OOuR9OtT2bRSeABTgYiYLJp7MoYkSil
5aJVmSQs4rPoyV8bE1EDr1Fd/1FVFZTckHy0J+/lwvGujlyLukqBQARjHAHls6ER
AgMBAAGjgbgwgbUwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4EFgQUm/IhmCFmTSSf
vOXRn2Q1R3KpXFwwHwYDVR0jBBgwFoAUm/IhmCFmTSSfvOXRn2Q1R3KpXFwwDgYD
VR0PAQH/BAQDAgEGMBkGA1UdEQQSMBCBDnJvb3RAbG9jYWxob3N0MDcGA1UdHwQw
MC4wLKAqoCiGJmh0dHA6Ly9hbm1hdC1wa2kvcGtpL3B1Yi9jcmwvY2FjcmwuY3Js
MA0GCSqGSIb3DQEBCwUAA4IBAQBL5KOebqndmIbeIMzutwwrMijr552LDqk3I2yu
QimR7oqypBOxEyutBxgaxKdhKxw1Qpz0DJZJa0gLZl2JU468LZr82HtcfOI6GDAH
13qxiD6h6sNrmmhnBE5TMbNw6R8V0ZPwalBR/R8/im2J1x0NQCNRcK6NuWFlruk5
cqEoMG4sJfunq65EUjtSOz2/0D52YLlzwjcETXDtGHUrQFs4Qj0RTZEgr62i+pan
YfkYebHTrzQVnOkdszf3QCxOITK9EmJaUxSZxHlaY76vnzVZArhmLITqRUYdVj1a
vl6yts7tJBGz9lBK61fAR9q4evRv8mgUFSgW7fkoX6ap8FdO
-----END CERTIFICATE-----
```

9.2. Pruebas de funcionamiento

Las pruebas de funcionamiento del servicio implementado pueden ser realizadas utilizando los comandos provistos por OpenSSL además de la interfaz web implementada para el chequeo y listado de sellos emitidos.

Para la emisión de un sello de tiempo, se utilizarán instrucciones del sistema operativo, como se muestra a continuación.

9.2.1. Pruebas mediante la línea de comando OpenSSL

Sellado de tiempo

Mediante el comando "openssl" utilizando los diversos parámetros soportados, es posible hacer una solicitud del sello de tiempo y obtener el sello de tiempo correspondiente.

```
openssl ts -query -data prueba -cert -sha1 | tee request.tsq | curl -s -S -H 'Content-
    Type: application/timestamp-query' --data-binary @- http://localhost/tsa -o
    response.tsr
```

Este comando, invoca a OpenSSL para obtener el sello de tiempo del archivo denominado prueba, almacena la solicitud del sello de tiempo en el disco del solicitante y por último la transmite mediante el protocolo HTTP a la TSA. La TSA devuelve el sello

de tiempo que es almacenado en el archivo denominado response.tsr. Para realizar esta prueba es necesario contar con la versión de OpenSSL con soporte de timestamping.

Verificación de sello de tiempo

```
openssl ts -verify -digest d136c1f587ee7e0c6a1d1ce66a3c368041f5bbdc -in /root/
    response.tsr -untrusted /root/server.pem  -CAfile /root/cacert.pem
```

```
Verification: OK
```

Nuevamente se invoca a la librería OpenSSL con diversos parámetros. El parámetro *digest* indica el hash del archivo a verificar, el parámetro *in* indica el archivo que contiene el sello de tiempo, el parámetro *untrusted* indica el certifica de la TSA y el parámetro *CAfile* indica el certificado raiz, que se utilizó para firmar el certificado de la TSA.

Si se intenta verificar un sello de tiempo haciendo uso de todos los mismos parámetros que la verificación exitosa, pero indicando un archivo que no corresponde con el resumen, la verificación va a dar fallida, debido a que el sello de tiempo no se corresponde con el archivo al que se hace referencia.

```
openssl ts -verify -digest d136c1f587ee7e0cea1d1ce66a3c368041f5bbdc -in /root/
    salidaPrueba.tsr -untrusted /root/server.pem  -CAfile /root/cacert.pem
```

```
Verification: FAILED
3076803244:error:2F064067:time stamp routines:TS_CHECK_IMPRINTS:message imprint
    mismatch:ts_rsp_verify.c:659:
```

Visualización de sello de tiempo

```
/usr/local/ssl/bin/openssl ts -reply -in response.tsr -text
```

```
Status info:
Status: Granted.
Status description: unspecified
Failure info: unspecified

TST info:
Version: 1
Policy OID: 1.1.2
Hash Algorithm: sha1
Message data:
    0000 - d1 36 c1 f5 87 ee 7e 0c-6a 1d 1c e6 6a 3c 36 80   .6....~.j...j<6.
    0010 - 41 f5 bb dc                                       A...
Serial number: 0x2A
Time stamp: Aug 30 22:06:26 2010 GMT
Accuracy: unspecified seconds, 0x01F4 millis, unspecified micros
Ordering: no
Nonce: 0xBCA16EDED3B50E1E
TSA: DirName:/C=AR/OU=UNLP/L=La Plata/CN=tsa.linti.unlp.edu.ar
Extensions:
```

Haciendo uso de la librería OpenSSL nuevamente, se visualiza en texto plano el contenido del sello de tiempo recibido.

Verificación de algoritmos soportados

Si se solicita un sello de tiempo a la TSA utilizando como algorítimo de hash MD5 y luego se visualiza el contenido del sello recibido, se obtiene el siguiente error:

```
openssl ts -query -data prueba -cert -md5 | tee request.tsq | curl -s -S -H 'Content-
    Type: application/timestamp-query' --data-binary @- http://localhost/tsa -o
    response.tsr
```

```
Status: Rejected.
Status description: Message digest algorithm is not supported.
Failure info: unrecognized or unsupported algorithm identifier

TST info:
Not included.
```

Con esto comprobamos que el algoritmo MD5 no es soportado por esta Autoridad de Sellado de Tiempo.

9.2.2. Pruebas utilizando el frontend web

Utilizando el frontend web desarrollado es posible realizar la verificación de los sellos de tiempo. Para ello, como ya he comentado, es necesario subir al servidor el sello de tiempo emitido, almacenado en un archivo con extensión .tsr.

En el caso que el sello se inválido, se visualizará una pantalla como la que observa en la figura 9.8; en este caso el sello de tiempo fue emitido por una Autoridad de Sellado que no corresponde con la del prototipo.
En el caso de la figura 9.9, el sello emitido es válido ya que fue emitido por la Autoridad de Sellado de Tiempo implementada para este trabajo. En ambos casos, se visualizan distintos campos informativos, que proporcionan al usuario información relacionada con el sello.

9.2.3. Accesos al sistema

El cuadro 9.2 con el que se concluye este capítulo, detalla los distintos accesos a los servicios proporcionados por el prototipo generado. En el mismo se encentran tanto los accesos al servicio de sellado de tiempo como los accesos al frontend web.

Servicio	URL
Acceso a la TSA por HTTP	http://tsa.linti.unlp.edu.ar/tsa
Acceso a la TSA port HTTPS	https://tsa.linti.unlp.edu.ar/tsa
Frontend público por HTTP	http://tesisalejandro.linti.unlp.edu.ar:81
Frontend público por HTTPS	https://tesisalejandro.linti.unlp.edu.ar:8443
Frontend de monitoreo por HTTP	http://tesisalejandro.linti.unlp.edu.ar:81/monitoreo
Frontend de monitoreo por HTTPS	https://tesisalejandro.linti.unlp.edu.ar:8443/monitoreo

Cuadro 9.2: Tabla con los accesos a los servicios brindados por el prototipo

Figura 9.8: Sello de tiempo inválido

Figura 9.9: Sello de tiempo válido

Capítulo 10

Seguridad del servicio

En este capítulo se explicarán los requerimientos y tareas realizadas en el marco de la seguridad y el monitoreo de los servicios y componentes que comprenden el prototipo de la Autoridad de Sellado de Tiempo. En primer término, se explicará el concepto de firewall o cortafuego y la implementación del mismo en la infraestructura implementada. Posteriormente, se hace referencia a la configuración realizada para proveer un acceso seguro al servicio y para concluir se especifican los diversos sistemas de monitoreo configurados para poder obtener información estadística y alertas relacionadas con todos los servicios involucrados.

En la figura 10.1 la interfaz de monitoreo desarrollada, la cual permite el acceso a los distintos servicios implementados.

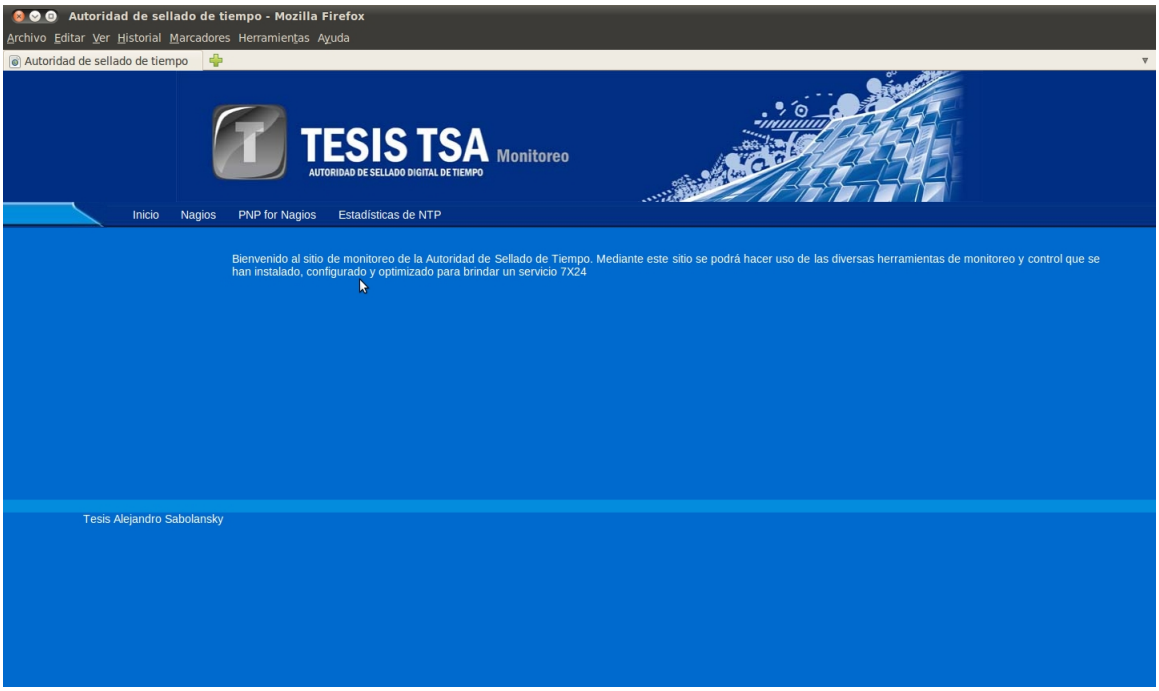

Figura 10.1: Interfaz de monitoreo

10.1. Instalación de firewalls

Un firewall o cortafuegos es un componente de un sistema o una red que está diseñado para bloquear el acceso no autorizado, permitiendo al mismo tiempo comunicaciones autorizadas. Se trata de un dispositivo o conjunto de dispositivos configurados para permitir, limitar, cifrar y descifrar el tráfico entre los diferentes ámbitos en base a un conjunto de normas y otros criterios definidos en el marco de una política de seguridad de la organización. Los cortafuegos pueden ser implementados en hardware o software, o una combinación de ambos.

Como parte de la implementación de una infraestructura que permita montar una autoridad de sellado de tiempo segura, se han configurado diversos Firewalls de red y de host para garantizar que los usuarios del servicio de sellado de tiempo digital puedan acceder únicamente a los servicios publicados.

En concreto, se ha utilizado el firewall provisto por Linux, conocido como iptables, basado en el framework disponible directamente en el núcleo del sistema operativo y se ha puesto en práctica una política restrictiva como metodología de filtrado.

10.2. Alternativas de acceso (HTTP / HTTPS)

El producto OpenTSA, que se integra con Apache, implementa como mecanismo de transporte el protocolo HTTP. Este protocolo no cifra la información que se envía en la comunicación entre el cliente y el servidor, por lo que la confidencialidad entre el solicitante del sello de tiempo y la Autoridad de sellado de tiempo, no se garantiza.

Para poder suplir dicha falencia, he configurado el acceso alternativo a la TSA mediante el protocolo HTTPS, utilizando la Autoridad de Certificación implementada para emitir el certificado digital necesario para el servidor web.

De esta forma, el usuario puede optar por cualquiera de los dos protocolos para poder interactuar con la TSA, tanto para la solicitud de sellos de tiempo como para el acceso a la interfaz de verificación y consulta desarrollada.

10.3. Monitoreo de componentes

En un servicio que debe estar disponible 7X24, es necesario contar con un conjunto de herramientas que permitan monitorear el servicio en forma constante, de manera tal que cualquier anomalía en alguno de los componentes de la arquitectura, pueda ser detectada y subsanada en forma inmediata.

Teniendo en cuenta la experiencia personal adquirida en el campo de monitoreo de redes y servicios, he seleccionado las herramientas más adecuadas y más aceptadas por la comunidad, y he configurado y optimizado las mismas con el objetivo de contar con la información necesaria para el análisis de comportamiento y monitoreo de disponibilidad de todos los componentes desarrollados.

Para ello, he configurado y optimizado diversas soluciones de software libre que se utilizan para el monitoreo de servicios:

- MRTG
- Nagios

- PNP for Nagios

Para restringir el acceso a estos servicios, he configurado la autenticación provista por Apache para asegurar que únicamente las personas autorizadas puedan acceder a las herramientas de monitoreo.

10.3.1. Monitoreo de servicios utilizando Nagios

Nagios[61] es un sistema de monitoreo de redes de código abierto ampliamente utilizado, que vigila los equipos (hardware) y servicios (software) que se especifiquen, alertando cuando el comportamiento de los mismos no sea el deseado. Se trata de un software que proporciona una gran versatilidad para consultar prácticamente cualquier parámetro de interés de un sistema, y generar alertas, que pueden ser recibidas por los responsables correspondientes mediante (entre otros medios) correo electrónico y mensajes SMS, cuando estos parámetros exceden de los umbrales definidos por el administrador de red. Nagios está licenciado bajo la GNU General Public License Version 2 publicada por la Free Software Fundation.

He configurado esta herramienta para el monitoreo de los servicios de conectividad, de las base de datos, del servidor web, y del servidor NTP, verificando en este último caso la precisión del reloj local, para evitar el desvío de 500 milisegundos que he fijado como máxima desviación posible para el servicio de sellado de tiempo.

Figura 10.2: Nagios

En la tabla que se encuentra a continuación, se muestran los servicios que se verifican para cada uno de los equipos.

Equipo	Servicio	Puerto
NodoCentral	Carga actual	-
NodoCentral	Chequeo de PostgreSQL Cluster	TCP 9999
NodoCentral	Chequeo de PostgreSQL Nodo	TCP 5432
NodoCentral	Chequeo del servicio TSA sobre HTTPS	TCP 443
NodoCentral	Chequeo del servicio TSA sobre HTTP	TCP 80
NodoCentral	Chequeo del Frontend sobre HTTPS	TCP 8443
NodoCentral	Chequeo del Frontend sobre HTTP	TCP 81
NodoCentral	Chequeo del Servicio de NTP	-
NodoSlave	Chequeo del Servicio de PostgreSQL Nodo	TCP 5432

En la siguiente imágen se visualiza la interfaz de monitoreo de Nagios configurada con los servicios implementados.

Host	Service	Status	Last Check	Duration	Attempt	Status Information
NodoSlave	Chequeo de Postgresql nodoSlave	OK	2010-07-10 14:59:58	5d 20h 30m 36s	1/4	POSTGRES_CONNECTION OK: DB "tsa" (host[redacted]) version 8.3.11
nodoCentral	Cantidad de usuarios	OK	2010-07-10 14:58:17	0d 20h 34m 19s	1/4	USERS OK - 0 users currently logged in
	Carga actual	OK	2010-07-10 14:59:03	0d 20h 33m 33s	1/4	OK - load average: 0.00, 0.00, 0.00
	Chequeo de Postgresql nodo Cluster	OK	2010-07-10 14:59:49	0d 20h 32m 47s	1/4	POSTGRES_CONNECTION OK: DB "tsa" (host:127.0.0.1) (port=9999) version 8.3.11
	Chequeo de Postgresql nodo1	OK	2010-07-10 15:00:35	0d 20h 32m 1s	1/4	POSTGRES_CONNECTION OK: DB "tsa" (host:127.0.0.1) version 8.3.11
	Chequeo del puerto 443 SSL	OK	2010-07-10 15:01:21	0d 20h 31m 15s	1/4	HTTP OK HTTP/1.1 200 OK - 324 bytes in 0.044 seconds
	Chequeo del puerto 81 Web publico	OK	2010-07-10 15:02:07	0d 20h 30m 29s	1/4	HTTP OK - HTTP/1.1 302 Found - 0.001 second response time
	Chequeo del puerto 8443 Web publico SSL	OK	2010-07-10 14:57:54	0d 20h 29m 42s	1/4	HTTP OK - HTTP/1.1 302 Found - 0.053 second response time
	Chequeo del servicio HTTP de la TSA	OK	2010-07-10 14:58:40	0d 20h 33m 56s	1/4	HTTP OK HTTP/1.1 200 OK - 324 bytes in 0.001 seconds
	Chequeo del servicio de NTP	OK	2010-07-10 14:59:26	0d 20h 33m 10s	1/4	NTP OK: Offset -1.978740329e-05 secs
	Espacio en disco	OK	2010-07-10 15:00:12	0d 20h 32m 24s	1/4	DISK OK
	Procesos totales	OK	2010-07-10 15:00:58	0d 20h 31m 38s	1/4	PROCS OK: 112 processes
	SSH	OK	2010-07-10 15:01:44	0d 20h 30m 52s	1/4	SSH OK - OpenSSH_5.1p1 Debian-5 (protocol 2.0)

Figura 10.3: Frontend de Nagios con los servicios configurados

Figura 10.4: MRTG

10.3.2. Monitoreo de NTP mediante MRTG

MRTG (Multi Router Traffic Grapher)[62] es una herramienta, escrita en C y Perl, que se utiliza para supervisar la carga de tráfico de interfaces de red. MRTG genera un informe en formato HTML con gráficas que proveen una representación visual de la evolución del tráfico a lo largo del tiempo.

Para recolectar la información del tráfico del dispositivo (habitualmente routers) la herramienta utiliza el protocolo SNMP (Simple Network Management Protocol). Este protocolo proporciona la información en crudo de la cantidad de bytes que han pasado por los mismos distinguiendo entre entrada y salida. Esta cantidad bruta debe ser transformada adecuadamente para poder realizar la generación de informes.

Además, permite ejecutar cualquier tipo de aplicación en lugar de consultar un dispositivo SNMP. Esta aplicación proporciona como salida dos valores numéricos que se corresponden a la entrada y salida.

Haciendo uso de estas características, esta herramienta ha sido utilizada de forma extensiva y adaptada para tratar información que no se adecúa a las medidas entrada/-salida como procesos.

Por esa razón, el autor de la herramienta decidió crear una segunda herrmienta RRDtool más flexible que permite almacenar cualquier tipo de datos. En las últimas versiones, MRTG utiliza RRDtool quedando restringida su funcionalidad a acceder a los dispositivos configurados para alimentar al RRDtool. La gestión de los datos y la generación de las gráficas se realizan mediante RRDtool.

Con MRTG he configurado el monitoreo de NTP para visualizar gráficamente la diferencia horaria entre UTC y el reloj configurado en el servidor de tiempo, con el fin

de detectar falencias en el servicio que se alejen de los 500 milisegundos soportados.

A continuación detallo el archivo de configuración de MRTG para la creación de la gráfica deseada:

```
WorkDir: /var/www/mrtg
WriteExpires: Yes

Title[^]: Analisis de átrfico


Target[tesisAle]: `perl /root/ntp1.pl localhost`
MaxBytes[tesisAle]: 100000
Title[tesisAle]: íEstadsticas de tiempo de la TSA - offset con respecto a  UTC
Options[tesisAle]: integer, gauge, nopercent, growright, nobanner
Colours[tesisAle]: Azul#0033FF, Rojo#FF0000, Azul#0033FF, Rojo#FF0000,
YLegend[tesisAle]: offset +/- us
ShortLegend[tesisAle]: microsegundos
LegendI[tesisAle]: offset microsegundos (-): 
LegendO[tesisAle]: offset microsegundos (+): 
Legend1[tesisAle]: Time offset in microsegundos (-)
Legend2[tesisAle]: Time offset in microsegundos (+)
PageTop[tesisAle]: <H1>Estadisticas del servidor de NTP</H1>
```

El componente principal de esta solución, es un script desarrollado en Perl que invoca a los comandos de NTP disponibles en el sistema operativo, con el cual se obtienen los diversos valores de sincronización, los cuales son manipulados y finalmente devueltos a MRTG para ser utilizados como datos de entrada en la generación de los gráficos.

```
# Espera el servidor como áparmetro.
# Retorna el primer valor como desplazamiento positivo y el segundo como negativo.
# Retorna los valores en microsegundos.
$ntp_str = `ntpq -c rv $ARGV[0]`;       # ejecuta "ntpq -c rv <servidor>"
$val = (split(/\,/,$ntp_str))[20];      #obtiene el offset
$val =~ s/offset=//i;                   # remueve el offset"
$val = int (1000 * $val);               # convierte a microsegundos
$nval = $val;                           # prepara el valor negativo
if ($val < 0){
$nval = -$nval;                         # crea el valor positivo
$val = 0;                               # asegura el 0 como retorno positivo
} else {
$nval = 0;                              # aseguro el o como retorno negativo
}
print "$nval\n";                        # retorna los valores
print "$val\n";
print "0\n";
print "$ARGV[0]\n"
```

En la figura 10.5, se observa la gráfica obtenida mediante MRTG mostrando el comportamiento de NTP a lo largo del tiempo; en la misma se puede observar el desvío en relación a UTC.

10.3.3. Análisis de comportamiento utilizando PNP4Nagios

PNP4Nagios[63] es un agregado para Nagios que básicamente, genera gráficas con los resultados de los análisis realizados por Nagios, con el objetivo de llevar un control más general del monitoreo de un determinado servidor o servicio en las últimas horas, días, semanas, meses o incluso años.

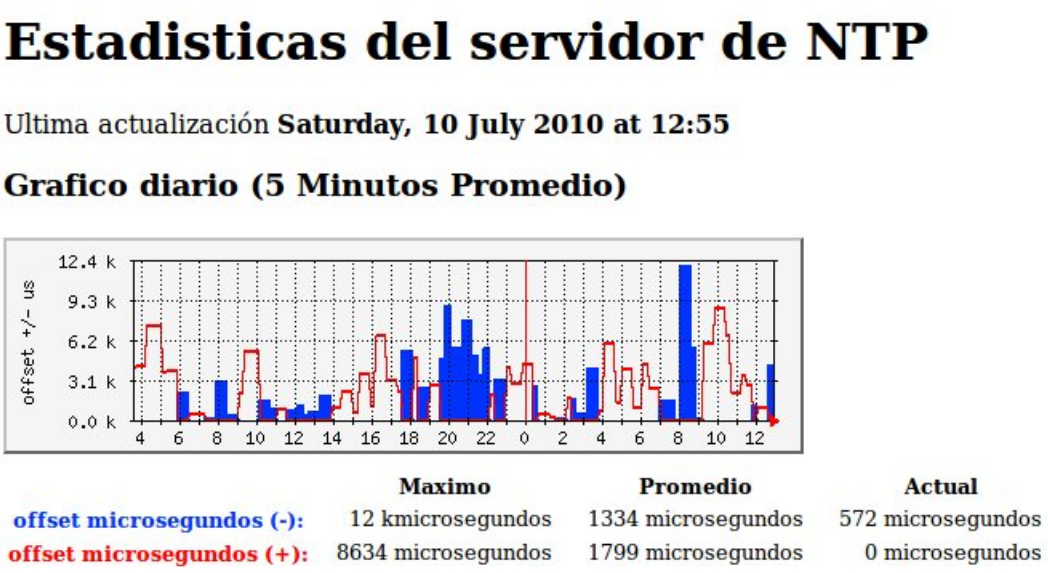

Figura 10.5: Grafico de NTP generada por MRTG

A fin de poder obtener estadísticas en forma sencilla sobre los diversos componentes del servicio de sellado de tiempo, he decidido utilizar esta herramienta. Para ello, he descargado, compilado, instalado y configurado este producto para poder integrarlo en el portal disponible mediante el perfil "Administrador" del Frontend Web generado.

En la figura 10.6 se observa el comportamiento del servicio brindado por la Autoridad de Sellado de Tiempo sobre el protocolo HTTP.

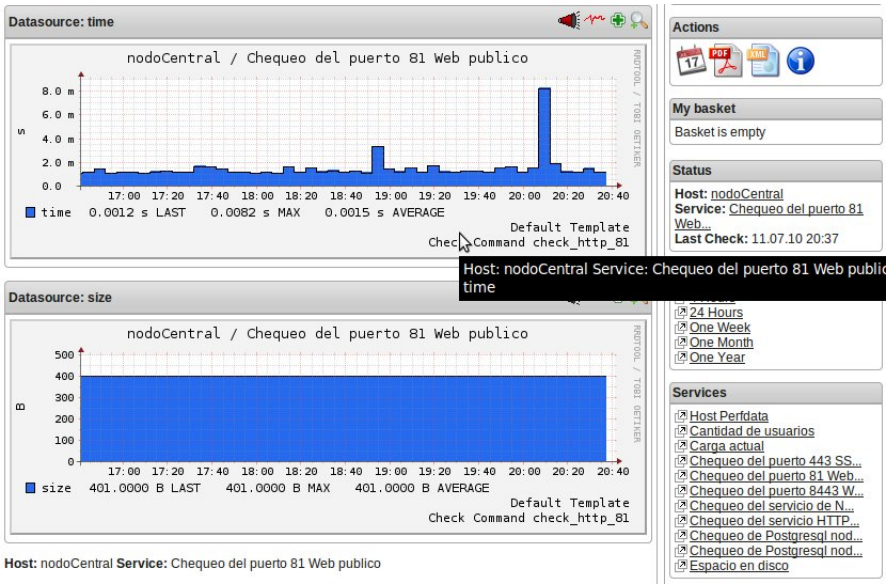

Figura 10.6: Captura de PNP4Nagios

Capítulo 11

Conclusiones

En este capítulo se van a desarrollar las conclusiones sobre el trabajo de tesis, como así también los trabajos que quedaron pendientes para su posterior investigación y desarrollo.

11.1. Conclusiones finales

Los objetivos planteados al comienzo del trabajo se cumplieron al lograr la puesta en funcionamiento del servicio, teniendo en cuenta los requerimientos de seguridad y monitoreo necesarios para brindar un servicio 7X24.

En primer término, quiero resaltar el carácter innovador de esta tesis, ya que desde el momento en que comencé a trabajar en la misma, he observado que no se han realizado gran cantidad de trabajos en el ámbito académico relacionados con este tema.

Por otra parte, me gustaría hacer hincapié en el aporte que genera este trabajo en la formación académica universitaria. Los tópicos desarrollados en esta tesis pueden ser incorporados como material de estudio en la cátedra de Seguridad y Privacidad en Redes, tanto en las carreras de grado como en la Maestría en Redes de Datos.

Por último, a partir de la experiencia adquirida durante el presente trabajo, es importante destacar los siguientes aspectos que considero relevantes en relación a la implementación del servicio:

- Resulta imprescindible la implementación del servicio de sellado digital de tiempo debido a que la firma digital no garantiza el instante de tiempo en que se ha realizado la firma.

- Al momento de implementar una Autoridad de Sellado de Tiempo, la misma debe enmarcarse en la normativa vigente tanto para la definición de las políticas y procedimientos como para la implementación del servicio en sí mismo.

- El servicio puede ser implementado en su totalidad con componentes open source aprovechando las ventajas que otorga este paradigma. Las herramientas utilizadas, son desarrollos sustentados por una gran comunidad de usuarios y programadores alrededor del mundo, lo que convierte a estos productos en software estable y confiable.

- Es necesario actualizar constantemente el servicio implementado de acuerdo al estado del arte de los algoritmos criptográficos y demás componentes involucrados en la solución.

11.2. Trabajos Futuros

Con respecto a las líneas de trabajo futuro, se han identificado las siguientes:

- Definición formal de la Política de Certificación y de la Declaración de Prácticas de Certificación de la autoridad de sellado digital de tiempo.
- Análisis de factibilidad para adaptar PKIGrid CA UNLP de manera tal que permita emitir certificados digitales para el servicio de sellado de tiempo brindado por la TSA implementada.
- Implementación de aplicaciones, prototipos o agregados a soluciones existentes que hagan uso del servicio de sellado de tiempo implementado.

Bibliografía

[1] Alfred J. Menezes, Paul C. Van Oorschot, Scott A. Vanstone, and R. L. Rivest. Handbook of applied cryptography, 1997.

[2] William Stallings. *Cryptography and Network Security: Principles and Practice*. Pearson Education, 2002.

[3] National Institute of Standards and Technology. *FIPS PUB 46-3: Data Encryption Standard (DES)*. October 1999. supersedes FIPS 46-2.

[4] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES — the Advanced Encryption Standard*. Springer-Verlag, 2002.

[5] Specification for the advanced encryption standard (aes). Federal Information Processing Standards Publication 197, 2001.

[6] Patrick Gallagher, Deputy Director Foreword, and Cita Furlani Director. Fips pub 186-3 federal information processing standards publication digital signature standard (dss), 2009.

[7] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[8] R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321 (Informational), April 1992.

[9] Marc Stevens, Alexander Sotirov, Jacob Appelbaum, Arjen K. Lenstra, David Molnar, Dag Arne Osvik, and Benne de Weger. Short chosen-prefix collisions for md5 and the creation of a rogue ca certificate. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 55–69. Springer, 2009.

[10] Secure Hash Standard. FIPS Pub 180-1. National Institute of Standards and Technology. 17 April 1995.

[11] http://www.sun.com/blueprints/0801/publickey.pdf. Public Key Infraestructure Overview.

[12] R. Housley, W. Ford, W. Polk, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. RFC 2459 (Proposed Standard), January 1999. Obsoleted by RFC 3280.

[13] http://www.rsa.com/rsalabs/node.asp?id=2124. Sitio oficial de RSA Laboratories sobre PKCS.

[14] http://physics.nist.gov/GenInt/Time/world.html. Escalas de tiempo.

[15] http://www.nmm.ac.uk/places/royal-observatory. Real Observatorio de Greenwich.

[16] http://www.nist.gov/physlab/div847/faq.cfm. FAQ de tiempo y frecuencias elaborado por el NIST.

[17] http://www.bipm.org. The International Bureau of Weights and Measures.

[18] http://www.hidro.gov.ar. Servicio de Hidrografía Naval.

[19] http://www.ign.gob.ar/. Instituto geográfico Nacional.

[20] Mohamed-Slim Alouini and Student Member Tss. Global positioning system: An overview.

[21] D.L. Mills. Network Time Protocol (NTP). RFC 958, September 1985. Obsoleted by RFCs 1059, 1119, 1305.

[22] D.L. Mills. DCNET Internet Clock Service. RFC 778 (Historic), April 1981.

[23] D.L. Mills. DCN Local-Network Protocols. RFC 891 (Standard), December 1983.

[24] D.L. Mills. Algorithms for synchronizing network clocks. RFC 956, September 1985.

[25] D. Mills. Network Time Protocol (Version 3) Specification, Implementation and Analysis. RFC 1305 (Draft Standard), March 1992. Obsoleted by RFC 5905.

[26] J. Postel. Daytime Protocol. RFC 867 (Standard), May 1983.

[27] J. Postel and K. Harrenstien. Time Protocol. RFC 868 (Standard), May 1983.

[28] C. Adams, P. Cain, D. Pinkas, and R. Zuccherato. Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). RFC 3161 (Proposed Standard), August 2001. Updated by RFC 5816.

[29] S. Santesson and N. Pope. ESSCertIDv2 Update for RFC 3161. RFC 5816 (Proposed Standard), April 2010.

[30] ISO/IEC JTC 1/SC 27, Time stamping services - Part 1: Framework, ISO ISO-18014-1, August 2008.

[31] ISO/IEC JTC 1/SC 27, Time stamping services - Part 2: Mechanisms producing independent tokens, ISO ISO-18014-2, December 2009.

[32] ISO/IEC JTC 1/SC 27, Time stamping services - Part 3: Mechanisms producing linked tokens, ISO ISO-18014-3, December 2009.

[33] ANSI X9.95-2005, Trusted Time Stamp Management and Security, 2005, URL: http://x9.org/.

[34] ETSI Technical Specification TS 101 861 V1.2.1. (2001-11). Time stamping profile. Note: copies of ETSI TS 101 861 can be freely downloaded from the ETSI web site www.etsi.org.

[35] D. Pinkas, N. Pope, and J. Ross. Policy Requirements for Time-Stamping Authorities (TSAs). RFC 3628 (Informational), November 2003.

[36] Trusted Time Stamp Standards: A Comparison And Guideline Of ANS X9.95; Information Assurance Consortium white paper, March 2007.

[37] Stuart Haber and W. Scott Stornetta. How to time-stamp a digital document. *Journal of Cryptology*, 3:99–111, 1991.

[38] Dave Bayer, Stuart Haber, and W. Scott Stornetta. Improving the efficiency and reliability of digital time-stamping. In *Sequences II: Methods in Communication, Security and Computer Science*, pages 329–334. Springer-Verlag, 1993.

[39] Ahto Buldas, Peeter Laud, Helger Lipmaa, and Jan Villemson. Time-stamping with binary linking schemes. In *In Advances on Cryptology (CRYPTO*, pages 486–501. Springer-Verlag, 1998.

[40] Ahto Buldas, Helger Lipmaa, and Berry Schoenmakers. Optimally efficient accountable time-stamping, 2000.

[41] Kaouthar Blibech and Alban Gabillon. A new timestamping scheme based on skip lists. In Marina Gavrilova, Osvaldo Gervasi, Vipin Kumar, C. Tan, David Taniar, Antonio Laganà, Youngsong Mun, and Hyunseung Choo, editors, *Computational Science and Its Applications - ICCSA 2006*, volume 3982 of *Lecture Notes in Computer Science*, pages 395–405. Springer Berlin / Heidelberg, 2006.

[42] http://www.opentsa.org. Sitio de OpenTSA.

[43] The OpenSSL Project. OpenSSL: The open source toolkit for SSL/TLS. www.openssl.org, April 2003.

[44] http://www.apache.org. Sitio Oficial de Apache.

[45] http://www.openevidence.org. Sitio de OpenEvidence.

[46] http://cordis.europa.eu/ist/home.html. Sitio donde se informa el fin del proyecto OpenEvidence.

[47] http://www.bouncycastle.org. Sitio de OpenTSA.

[48] http://www.digistamp.com/toolkitDoc/MSToolKit.htm. Sitio de Digistamp.

[49] http://www-03.ibm.com/systems/power/software/aix/index.html. Sitio Web de IBM relacionado con AIX.

[50] http://www.oracle.com/us/products/servers-storage/solaris/index.html. Sitio Web de Oracle relacionado con Solaris.

[51] http://www.freebsd.org. Sitio del Proyecto FreeBSD.

[52] http://www.openbsd.org. Sitio del Proyecto OpenBSD.

[53] http://www.netbsd.org. Sitio del Proyecto NetBSD.

[54] http://www.gnu.org. Página web de GNU.

[55] http://www.debian.org. Sitio Oficial de Debian.

[56] http://www.mysql.org. Sitio Oficial de MySQL.

[57] http://www.postgresql.org. Sitio Oficial de PostgreSQL.

[58] http://www.firebirdsql.org. Sitio Oficial de Firebird.

[59] http://www.pkigrid.unlp.edu.ar. Sitio del proyecto PKIGrid UNLP.

[60] Valeria Bertacco Andrea Pellegrini and Todd Austin. Fault-based attack of rsa authentication. University of Michigan.

[61] http://www.nagios.org. Sitio Oficial de Nagios.

[62] Tobias Oetiker and Traffic Grapher. Mrtg – the multi router.

[63] http://docs.pnp4nagios.org. Sitio Oficial de PNP4Nagios.