MAESTRÍA EN AUDITORIA DE TECNOLOGÍA DE LA INFORMACIÓN

ASEGURAMIENTO DE ARCHIVOS DIGITALES MEDIANTE UN SELLADO DE TIEMPO INDEPENDIENTE DE UNA AUTORIDAD CERTIFICADORA

SECURING DIGITAL FILES BY USING TIMESTAMPING THAT DOES NOT DEPEND ON A CERTIFICATION AUTHORITY

Propuesta de artículo presentado como requisito para la obtención del título:

Magíster en Auditoría de Tecnologías de la Información

Por el estudiante:
Romel André VERA CADENA

Bajo la dirección de:
Rayner Stalyn DURANGO ESPINOZA

Universidad Espíritu Santo
Maestría en Auditoría de Tecnología de la Información
Samborondón - Ecuador
Septiembre del 2017

ASEGURAMIENTO DE ARCHIVOS DIGITALES MEDIANTE UN SELLADO DE TIEMPO INDEPENDIENTE DE UNA AUTORIDAD CERTIFICADORA

SECURING DIGITAL FILES BY USING TIMESTAMPING THAT DOES NOT DEPEND ON A CERTIFICATION AUTHORITY

Romel André VERA CADENA[1]
Rayner Stalyn DURANGO ESPINOZA[2]

Resumen

El sellado de tiempo confiable se emplea para certificar que un documento informático se encuentre íntegro desde la fecha de su registro, sin embargo, el modelo actual de sellado de tiempo depende de autoridades certificadoras, las cuales están sujetas a la confianza y en la actualidad han existido problemas de confianza que han causado diversos daños que van desde la suplantación de identidad hasta la propagación de malware. Por consecuencia, el objetivo del presente estudio consiste en crear un esquema alternativo de sellado de tiempo que permita asegurar archivos digitales mediante un sellado de tiempo que no dependa de una autoridad certificadora. Para ello, se basó en una metodología explicativa en la que se analizaron casos de problemas de confianza con la finalidad de superar el problema de la confianza mediante la comprobación de integridad empleando la función hash. A partir de lo analizado, se verifica la posibilidad de emplear un esquema alternativo cuando se requiere evitar problemas de confianza, puesto que excluye a las autoridades certificadoras del esquema de sellado de tiempo.

Palabras clave: Sellado de Tiempo, Seguridad, Confianza, Certificado Digital, Autoridad Certificadora.

Abstract

Trusted timestamping is the process used to certify that a computer document is not altered from the date it was registered; however, the current timestamping procedure depends on a certifying authority, which is subject to trust, and currently there have been trust problems that have caused various damages, ranging from identity theft to the spread of malware. Consequently, the objective of this paper is to create an alternative time-stamping scheme that allows digital files to be secured by means of a time stamp that does not depend on a certification authority. To do this, the study was based on an explanatory methodology in which three cases of trust problems were analyzed in order to overcome the trust problem by checking the integrity using the hash function. Based on the analysis, the possibility of using the alternative scheme when it is necessary to avoid trust problems since it excludes the certifying authorities from the time-stamping scheme.

Key words: Timestamping, Security, Trust, Digital Certificates, Certificate Authority.

[1] Estudiante de Maestría en Auditoría de Tecnología de Información, Universidad Espíritu Santo – Ecuador. E-mail romel.vera.cadena@gmail.com
[2] Magíster en Sistemas de Información Gerencial. Ecuador. E-mail rdurango@uees.edu.ec

INTRODUCCIÓN

El procesamiento de datos digitales ha traído muchos beneficios a la forma en que se crea y se organiza la información, en forma de imágenes, documentos, archivos de audio o video. La conversión de dicha información digitalizada proporciona capacidades como el almacenamiento y recuperación, procesos de búsqueda eficientes y transmisión de datos de manera inmediata sin importar distancias. (Cassany, 2000)

Sin embargo, los datos digitales también traen consigo un problema importante, como es la facilidad del copiado y manipulación de la información, sin dejar evidencia alguna. Por lo que ha surgido la necesidad de contar con tecnologías capaces de salvaguardar la integridad y la autenticidad de los registros digitales; en particular si tales registros están sujetos a pruebas jurídicas o éticas. La preservación de la integridad de los registros digitales es particularmente importante cuando están sujetos a un posible ataque criminal. (Voutssas, 2010)

En consecuencia, en las organizaciones es necesario contar con herramientas que permitan comprobar o certificar la integridad de documentos informáticos, es decir, que deben tener la capacidad de determinar si un documento fue modificado sin autorización previa con la finalidad de evitar fraudes y por efecto perjuicios en la información. (Trejo, Domenech, & Ortíz, 2016)

En el caso de los documentos gubernamentales que se manejan con herramientas de editor de textos u hojas de cálculo, hay aplicaciones que ayudan a cifrarlos, pero esto puede complicar el manejo de los archivos, cuando solamente se desea que no sean adulterados. (Stone, 2016) (Armstrong & Mayo, 2009)

El objetivo general del presente artículo es crear un esquema alternativo de sellado de tiempo que permita asegurar archivos digitales mediante un sellado de tiempo que no dependa de una autoridad certificadora.

Con la finalidad de justificar el objetivo del presente artículo, se denotará el problema de la confianza y sus repercusiones en los certificados digitales y en las autoridades certificadoras.

Para lograr el objetivo planteado, a partir de los problemas encontrados se procederá a diseñar un esquema alternativo que se llamará Simple Hashing, la cual se demostrará por medio de aplicación SaaS/ScuS, finalizando con la comparación de soluciones similares.

MARCO TEÓRICO

SITUACIÓN ACTUAL EN LA INTEGRIDAD DE DOCUMENTOS Y SU PROBLEMÁTICA

Actualmente existe un modelo estándar de sellado de tiempo confiable para la verificación de integridad en los documentos informáticos, y estos requieren del uso de certificados digitales. (Une, 2001)

El sellado de tiempo confiable hace uso de certificados digitales sin embargo los certificados digitales están sujetos a la confianza, y la confianza se basa en dos principios:

1. La Autoridad Certificadora (CA) no debería emitir certificados a usuarios inapropiados.
2. Los usuarios no deben agregar ningún CA inapropiado a su lista de CAs de confianza.

Este concepto de confianza se encuentra en debate debido a que existen grandes organizaciones e instituciones que no necesariamente son de confianza. (Oppliger, 2014) (Roosa & Schultze, 2013)
En una investigación de la Universidad de Maryland publicada en la ACM reveló que, de 111 certificados usados para firmar malware, 75 certificados fueron comprometidos por mal manejo por parte de los usuarios finales, y 27 certificados fueron generados para creadores de malware por parte de las Autoridades Certificadoras debido a fallas en los procesos de verificación. (Kim, Kwon, & Dumitraş, 2017)

LOS CERTIFICADOS EMITIDOS Y LOS PROBLEMAS EN ORGANIZACIONES

Es posible conseguir certificados digitales usando cualquier nombre y en casos hasta el mismo dominio. Por ejemplo: LetsEncrypt emitió 15,270 certificados que contienen la palabra “Paypal” en el nombre del dominio. (Brenner, 2017)

También, como se indica en la tabla 1, existen casos de empresas conocidas a nivel mundial

que han tenido este tipo de problema. (Specter, 2016)

Tabla 1. Lista parcial de empresas que han tenido problemas de confianza

Empresa	Por medio de	Caso
Comodo	Diginotar	Paypal
Google	China Internet Network Information Center (CNNIC)	Google - CNNIC
Symantec	Varios CAs de Symantec	Google Chrome EV & Symantec
Kaspersky Lab	Foxconn	Stuxnet
Wosign, StartCom	Gobierno de China	Google vs China

Función hash

La función hash conocida también como funciones de resumen o message digest en inglés, son algoritmos de cálculo que, a partir de una entrada: texto, contraseña o archivos, crean una salida alfanumérica única, de modo que solo es posible volverla a crear si se utiliza exactamente la misma entrada (texto, contraseña o archivos). El objetivo de esta función es proveer una herramienta para verificar que un determinado texto o archivo no se encuentre manipulado sin la debida autorización. (Johnson, Badger, Waltermire, Snyder, & Skorupka, 2016)

Firma digital / firma electrónica

Las firmas digitales o electrónicas conocidas en inglés como Digital Signatures sirven para asegurar el origen del documento y la integridad de los datos del documento. La firma digital asegura que la persona que firma no pueda rechazar su propio documento a esto se le conoce como “No Repudio”. (Barker & Roginsky, 2015)

Para garantizar la seguridad de las firmas digitales es necesario que éstas sean únicas, infalsificables, verificables, innegables, viables. (Shirey, 2007)

Shirey (2007) indica que las propiedades de la firma digital son las siguientes:

Únicas: Las firmas deben poder ser generadas solamente por el firmante.

Infalsificables: Las firmas deben ser computacionalmente seguras.

Verificables: Las firmas deben ser fácilmente verificables por los receptores de las mismas.

Innegables o No repudiables: El firmante no debe ser capaz de negar su propia firma.

Viables: Las firmas deben ser fáciles de generar por parte del firmante.

En el marco legislativo ecuatoriano la firma digital se define como el conjunto de datos en forma electrónica, consignados junto a otros o asociados con ellos, que pueden ser utilizados como medio de identificación del firmante. (Banco Central del Ecuador, 2011)

Certificado Digital

Un certificado digital es un archivo informático generado por una entidad de servicios de certificación que asocia unos datos de identidad a una persona física, organismo o empresa confirmando de esta manera su identidad digital en Internet. El certificado digital es válido principalmente para autenticar a un usuario o sitio web en internet, por lo que es necesario, la colaboración de un tercero que sea de confianza para cualquiera de las partes que participe en la comunicación. El nombre asociado a esta entidad de confianza es Autoridad Certificadora pudiendo ser un organismo público o empresa reconocida en Internet. (Kuhn, Hu, Polk, & Chang, 2001)

Sellado de tiempo confiable

El sellado de tiempo confiable o “trusted timestamping” en inglés, es un mecanismo en línea que permite demostrar que una serie de

datos han existido y no han sido manipulados desde un instante específico en el tiempo. Este protocolo se describe en el RFC 3161 y está en el registro de estándares de Internet. (Zuccherato, Cain, Adams, & Pinkas, 2001) (ANSI, 2016)

Autoridad de sellado de tiempo
Una autoridad de sellado de tiempo (TSA) actúa como tercera parte de confianza testificando la existencia de una serie de datos en una fecha y hora concretos. (Une, 2001)

Software como un Servicio "ScuS / SaaS"
Software como un Servicio, abreviadamente ScuS (del inglés: Software as a Service, SaaS), es un modelo de distribución de software donde el soporte lógico y los datos que maneja se alojan en servidores centralizados en una compañía. (Turner, Budgen, & Brereton)

METODOLOGÍA

El presente método de la investigación es explicativo:

Figura 1. Metodología

Se procederá a describir el proceso de sellado de tiempo confiable para posteriormente revisar los casos más populares de problemas de confianza en los certificados digitales.

El alcance de la investigación abarca los problemas de confianza tomando en cuenta los tres casos de más impacto obtenidos por el índice del momento de popularidad de los resultados de Google del día martes 28 de marzo del 2017.

Luego se graficará en qué parte del esquema del modelo de sellado de tiempo afecta el problema de la confianza, con el objetivo de reestructurar el esquema y poder quitar la dependencia externa de la Autoridad Certificadora.

Posteriormente, se desarrollará una aplicación SaaS/ScuS demo en la que se hace uso del sellado de tiempo independiente de una autoridad certificadora.

Finalmente, se creará un cuadro comparativo de soluciones similares tomando en cuenta la verificación de la integridad, la verificación de la fecha, la dependencia de los CA, si es un servicio o no, y si es comercial o no.

Para la elaboración de la aplicación SaaS/ScuS demo se usará el sistema operativo Ubuntu 16.04 LTS, para el frontend se usará HTML5, CSS3, Javascript y los componentes jQuery, Datatables, Bootstrap, Responsive, y Select. Para el backend se usará Apache2, PHP7 y MySQL 5.

Se han escogido estas plataformas de desarrollo porque sus licencias permisivas consienten desarrollar software de código abierto por lo tanto no está sujeto a ninguna obligación con un tercero.

SELLADO DE TIEMPO CONFIABLE / TRUSTED TIMESTAMPING

Según el estándar RFC 3161 se puede obtener un sello de tiempo por medio de un tercero de confianza con el fin de probar la integridad de datos tales como: contratos, investigaciones, bitácoras médicas, documentos financieros, etc.

Proceso de Sellado de Tiempo Confiable

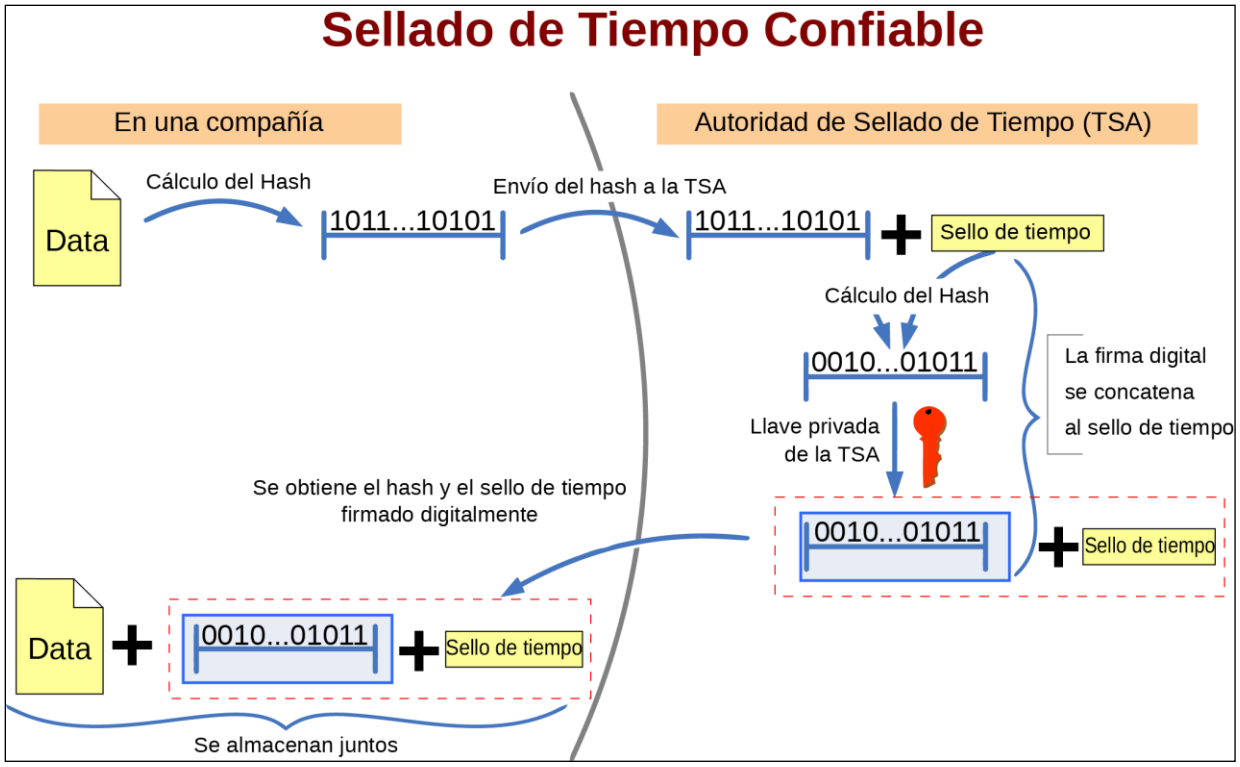


Figura 2. Sellado de Tiempo Confiable
Fuente: Bart Van den Bosch, 2004

En el proceso de sellado de tiempo confiable se describen los siguientes pasos:

1. Un usuario quiere obtener un sello de tiempo para un documento electrónico que él posee.
2. Un resumen digital “técnicamente un hash” se genera para el documento en el ordenador del usuario.
3. Este resumen forma la solicitud que se envía a la autoridad de sellado de tiempo (TSA).
4. La TSA genera un sello de tiempo con esta huella, la fecha y hora obtenida de una fuente fiable y la firma electrónica de la TSA.
5. El sello de tiempo se envía de vuelta al usuario.
6. La TSA mantiene un registro de los sellos emitidos para su futura verificación.

Y para la verificación:

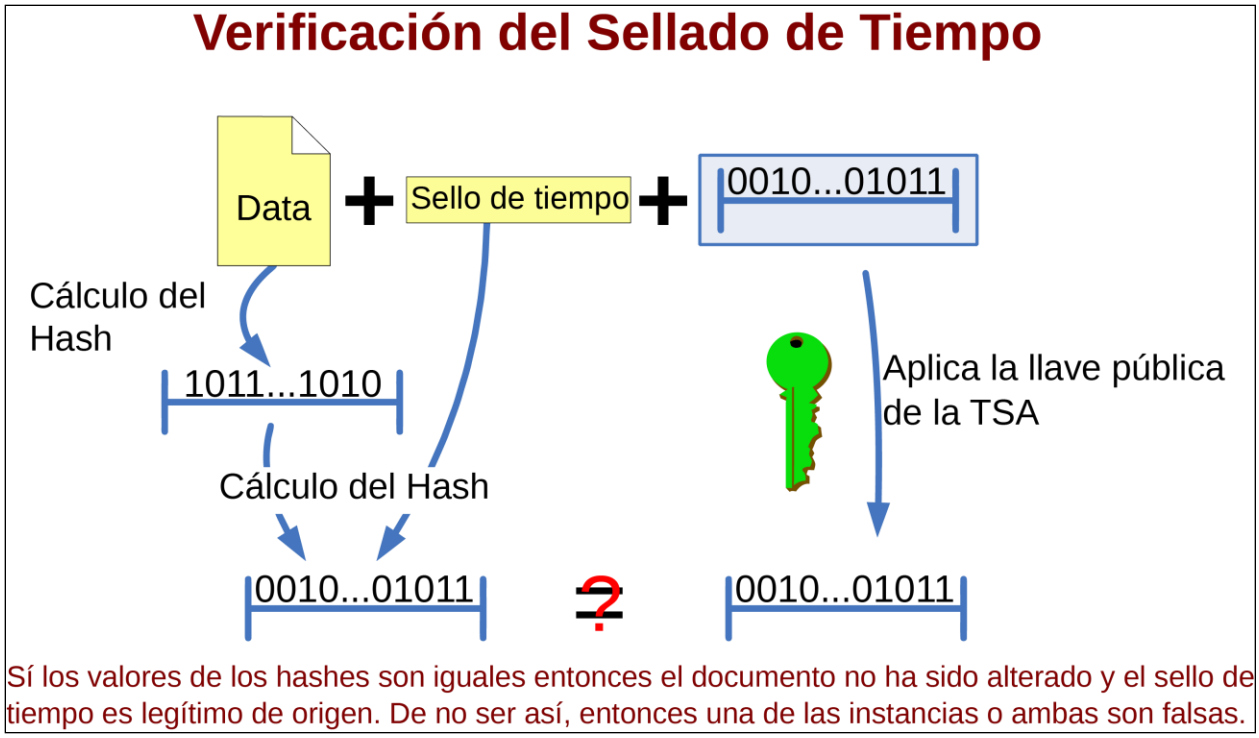


Figura 3. Verificación del Sellado de Tiempo
Fuente: Tsuruya, 2012

1. Se calcula el hash de la información original.
2. Se adjunta el sellado de tiempo otorgado por la TSA.
3. Se calcula nuevamente un hash de la información anterior. Llamemos a este último hash A.
4. El hash A es comparado con el hash B dentro del mensaje firmado por la TSA para poder comprobar que son idénticos, evidenciando que el sellado de tiempo y el mensaje no fueron alterados y que éste fue realmente publicado por la TSA. Si esto no ocurre, entonces el sellado de tiempo fue alterado o bien el sellado de tiempo no fue emitido por la TSA.

PROBLEMAS DE CONFIANZA, CASOS DE ESTUDIO

Los certificados digitales son la base fundamental de la confianza en línea. Estos certificados digitales se comparan con las firmas, y normalmente se confía en un documento porque tienen una firma o lo respalda una autoridad de certificación en la que se confía. Es decir que los certificados digitales son una reproducción de un modelo simple que ocurre en la vida real. (Lax, Buccafurri, & Caminiti, 2015)

Los incidentes que involucran certificados digitales y a las autoridades certificadoras (CA) no siempre son claros o notables para los usuarios finales. Sin embargo, los administradores de TI, los desarrolladores de software y otros profesionales de la seguridad de la información necesitan comprender estos problemas para que los riesgos puedan gestionarse adecuadamente. (Laurie, 2014) (Specter, 2016)

Existen numerosos casos en los que se ha perdido la confianza en las Autoridades Certificadoras (CAs). (Kim, Kwon, & Dumitras, 2017)

Google bloquea certificados de CAs de china WoSign y StartCom

En agosto 17 del 2016 la compañía GitHub había notificado a Google que la autoridad certificadora WoSign había emitido un certificado base a nombre de GitHub sin previa autorización, después de este reporte Google con la colaboración de Mozilla descubrieron algunos

casos de WoSign en las que emitía certificados no autorizados.
Como resultado Google procedió a eliminar progresivamente de la lista de confianza del navegador Google Chrome a todos los elementos pertenecientes a WoSign y StartCom. En el año 2016 Apple y Mozilla ya habían dejado de confiar en WoSign y StartCom debido a dificultades técnicas que tenían con éstas compañías. Adicionalmente Mozilla había encontrado que WoSign había adquirido el CA StartCom y esto nunca fue notificado a ninguna compañía pese a las políticas establecidas entre ellas. (Corser, 2015)

Symantec pierde confianza

Desde octubre del año 2015 la empresa Symantec ha estado perdiendo gradualmente confianza, debido a problemas de seguridad por la forma en que emite sus certificados digitales y anteriormente Symantec había prometido solucionar el problema, sin embargo, Google reporta que no ha sido el caso y que va a dejar de confiar en los certificados del tipo EV “de alta seguridad” que genera Symantec. Esta medida afecta a otras marcas que controla Symantec como son Geotrust, VeriSign, y Thawte.
Como contramedida frente a estos problemas Google va a asignar fechas de expiración temprana a los certificados de Symantec. En el navegador Google Chrome para su versión 62 que saldrá en octubre del 2017 va a fijar en los certificados digitales de Symantec una expiración de 15 meses a partir de su fecha de vigencia. (Goodin, 2017)

Variante de Stuxnet infecta Kaspersky por medio de certificados digitales Foxconn

En el año 2015 el malware Duqu 2.0 infectó la red corporativa de la empresa Kaspersky Lab creadora del anti-virus Kaspersky. El malware había pasado inadvertido debido a que estaba firmado con un certificado digital perteneciente a la compañía Foxconn que se encarga de manufacturar componentes electrónicos para iPhone, Xbox, Laptops Acer, Laptops Dell, entre otros productos conocidos. (Goodin, 2015)
En el sistema operativo Windows para poder instalar controladores o drivers en inglés, es necesario que los drivers se encuentren firmados con un certificado digital. Foxconn usa drivers firmados para ciertos componentes que fabrica y de alguna forma los atacantes se adueñaron de los certificados digitales para luego firmar el malware como un driver legítimo de Foxconn. Duqu 2.0 se había instalado en firewalls, puertas de enlace y otros servidores que tenían enlace directo a internet, nunca se llegó a conocer el verdadero propósito de los creadores del Duqu 2.0 aunque en los análisis de Kaspersky se refleja que los creadores tenían como objetivo principal las empresas que fabrican hardware. (Kaspersky, 2015)

ANÁLISIS DEL PROBLEMA DE LOS CASOS

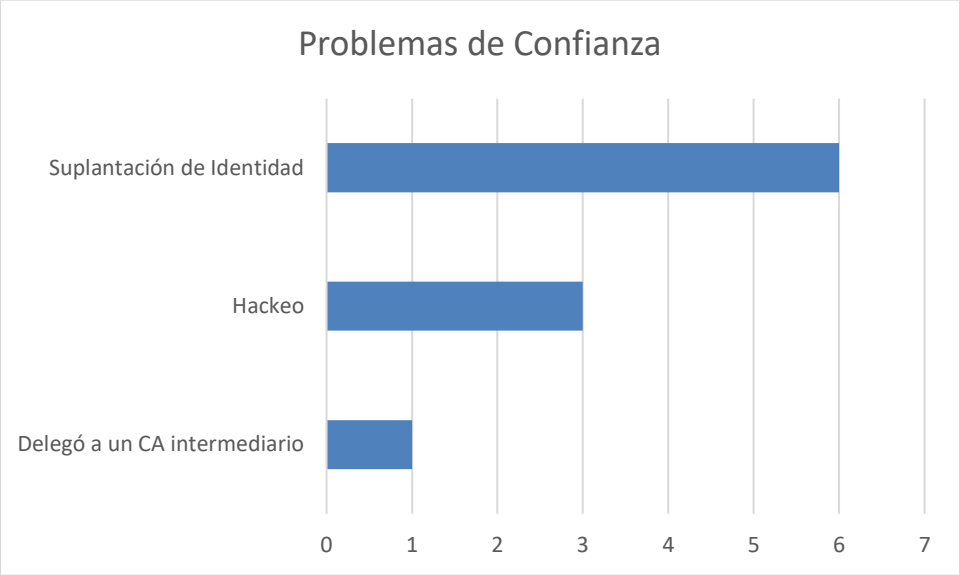
Los certificados digitales dependen mucho de la seguridad de la organización que los maneja y en el caso de los equipos de computación, laptops, PCs, servidores, celulares, entre otros equipos en su operación están sujetos a cadenas de confianza, cada aplicación, servicio o librería confía en uno o más CAs, por ejemplo: En un celular Samsung Galaxy Note es posible encontrar aproximadamente 300+ aplicaciones y servicios que dependen de certificados digitales para diversas funciones que van desde una conexión segura hasta la comprobación de integridad de aplicaciones al revisar una firma digital. (Blaich, 2015)

Es posible encontrar en Google noticias y estudios de problemas de confianza en CAs. En este sentido, con la finalidad de apoyar el análisis de los casos se elaboró una tabla a partir de los resultados de Google del día martes 28 de marzo del 2017.

Tabla 2. Muestra de 10 elementos que han tenido problemas de confianza

Año	Compañía Afectada	Por parte	Tipo	Problema	Motivo	Detalle Adicional
2001	Microsoft	VeriSign	CA	Suplantación de Identidad	Insuficiente validación	El atacante se hizo pasar por un empleado de Microsoft
2008	Microsoft Live	Thawte	CA	Suplantación de Identidad	Falta de controles de seguridad en la validación de dominios de correo	La dirección de correo registrada fue: "sslcertificates@live.com"
2011	Google Mozilla Microsoft Live Yahoo Microsoft Skype	Comodo	CA	Suplantación de Identidad	Insuficiente validación	Comodo delegó la validación a compañías revendedoras cuando esa función crítica debía hacerla el CA principal
2011	Google Numerosas compañías	Diginotar	CA	Hackeo	Prácticas de seguridad pobres Software Desactualizado	Se usó un certificado de Google para interceptar tráfico de datos de ciudadanos Iraníes.
2014	Google Yahoo	NICCA	CA	Hackeo	Se desconoce el motivo	Google indicó que NICCA nunca reportó los certificados comprometidos.
2015	Numerosas compañías	Symantec	CA	Suplantación de Identidad	Insuficiente validación	Emitir certificados sin previa autorización
2015	Kaspersky	Foxconn	Usuario Final	Suplantación de Identidad	Mal manejo de los certificados	Malware Duqu 2.0
2015	Google Numerosas compañías	CNNIC	CA	Delegó a MCS Holdings como CA intermediario	Incumplimiento de buenas prácticas al verificar condiciones de CA intermediarios	Permitía a cualquier usuario conseguir certificados para cualquier dominio
2016	Github	WoSign	CA	Suplantación de Identidad	Insuficiente validación	Emitir certificados sin previa autorización
2016	Numerosas compañías	StarCom	CA	Hackeo	Incumplimiento de buenas prácticas al desarrollar software y APIs	Permitía a los atacantes conseguir certificados para cualquier dominio

Cuadro Estadístico 1. Problemas de Confianza



En el cuadro estadístico 1 se puede apreciar que de los 10 elementos de la tabla 2, seis son problemas por suplantación de identidad, tres son problemas por hackeo, y un problema por un CA intermediario. Según Michael Alan Specter del MIT indica que los problemas de confianza que envuelven a las CAs se debe a la interacción humana entre usuarios e instituciones. Los usuarios no están preparados debidamente para realizar decisiones basadas en la confianza. (Specter, 2016)

Círculo de confianza

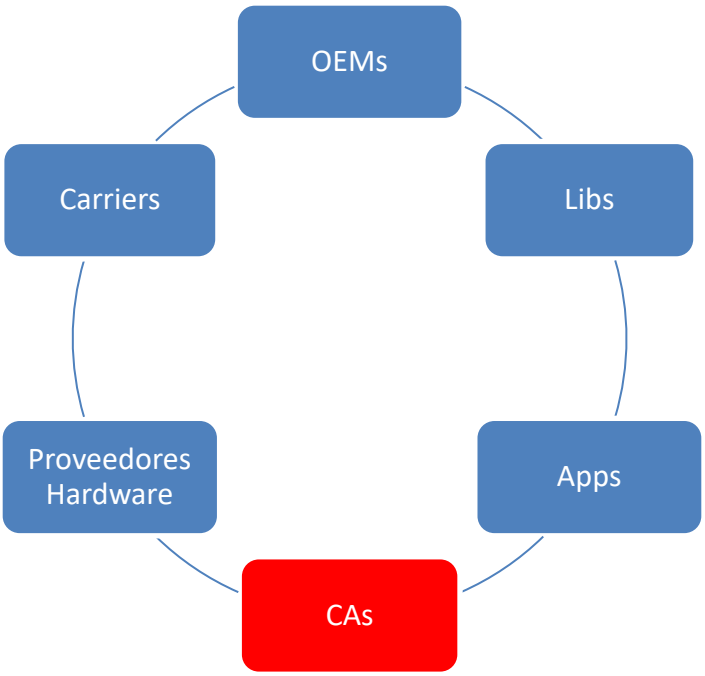


Figura 4. Círculo de Confianza
Fuente: Andrew Blaich, 2015

Un mal manejo de un certificado digital puede tener varias consecuencias tales como la suplantación de identidad, fraude en documentos, intercepción de comunicaciones, difusión de malware, y sabotaje. (Paganini, 2014) (Alrawi & Mohaisen, 2016)

GRÁFICO DEL PROBLEMA DE CONFIANZA EN EL PROCESO DE SELLADO DE TIEMPO CONFIABLE

En el proceso de sellado de tiempo confiable el problema de la confianza afectaría al paso número 4:

4. La TSA genera un sello de tiempo con esta huella, la fecha y hora obtenida de una fuente fiable y la firma digital de la TSA.

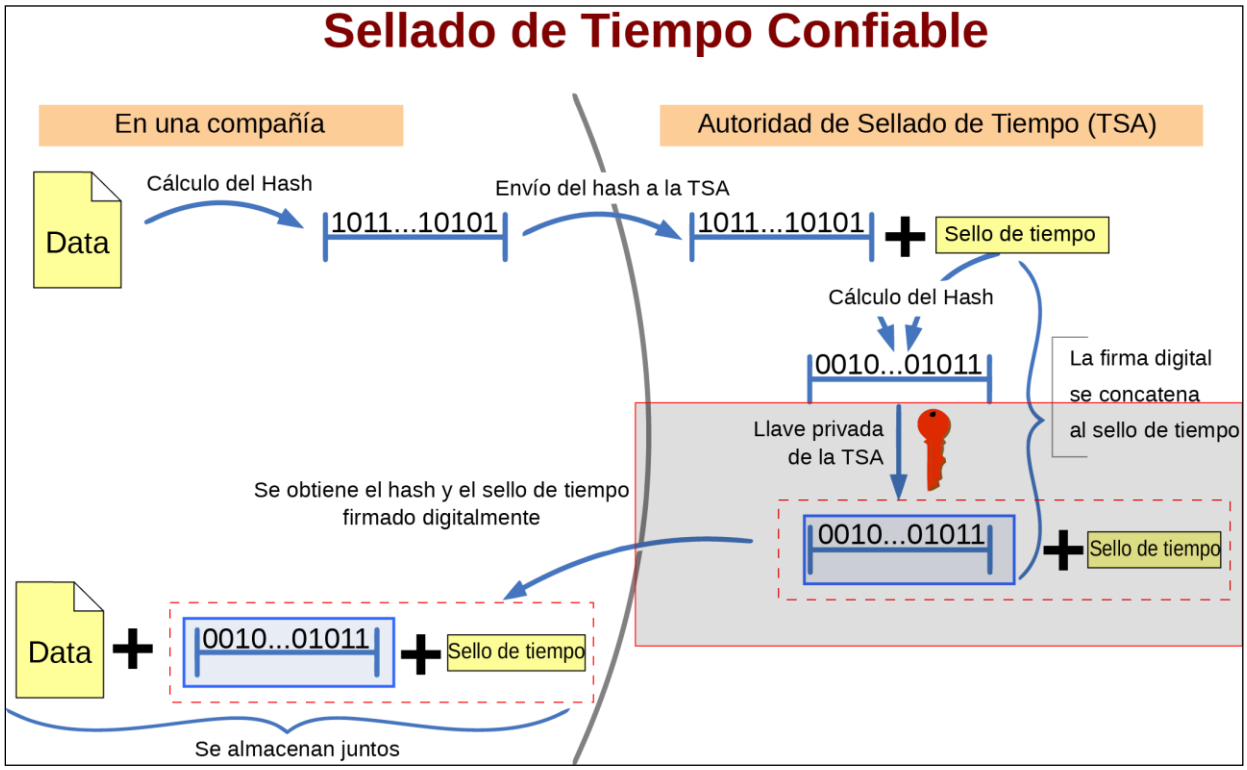


Figura 5. Gráfico del problema de confianza en el proceso de sellado de tiempo confiable
Fuente: Bart Van den Bosch, 2004

SELLADO DE TIEMPO INDEPENDIENTE DE UNA AUTORIDAD CERTIFICADORA

El aseguramiento de archivos digitales mediante un sellado de tiempo independiente de una autoridad certificadora se llama “Simple Hashing” que es una simplificación del esquema actual de sellado de tiempo confiable que no requiere de un certificado digital ni genera un archivo de comprobación puesto que la comprobación se hace directamente en el servidor.

Beneficios del Simple Hashing

- No requiere un certificado de una Autoridad Certificadora. (No dependería de la confianza de una autoridad externa)
- El usuario no almacena o lleva archivos adicionales.
- Resultado inmediato en pantalla.
- Menos complejidad de uso al evitar el manejo de certificados digitales.
- Menos complejidad en la implementación debido a la simplificación del proceso.

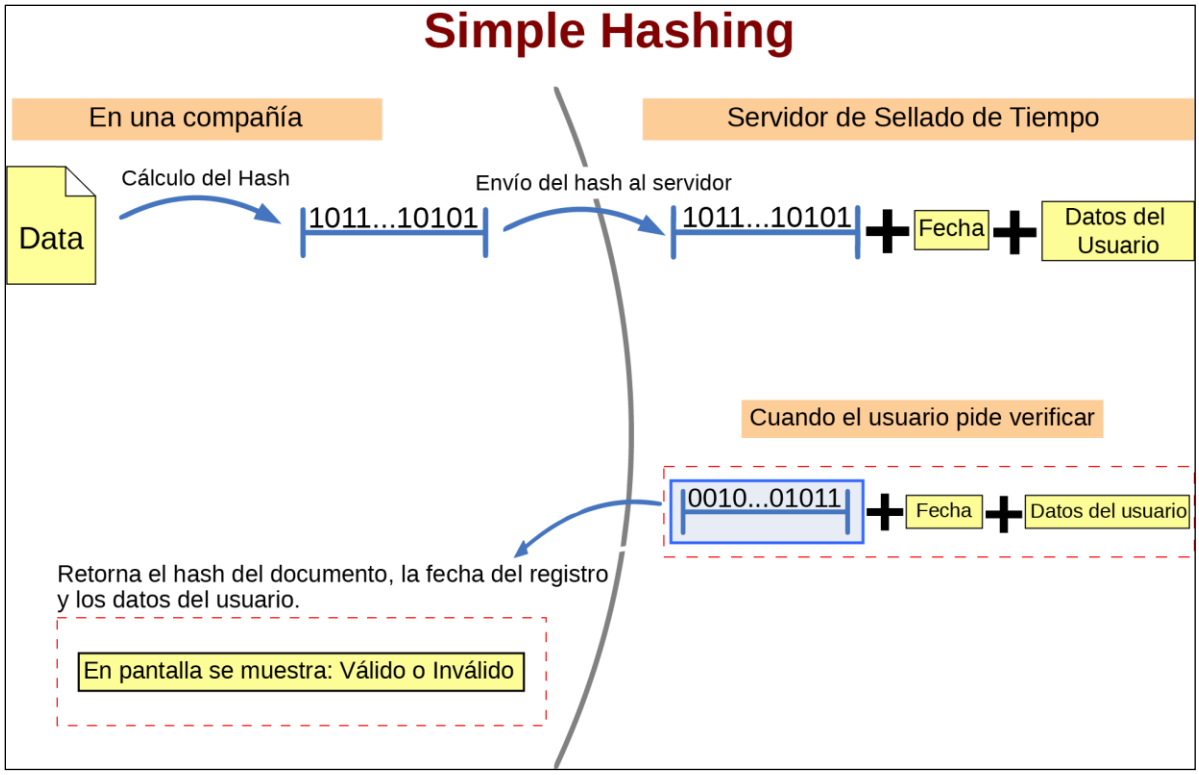


Figura 6. Simple Hashing

En el proceso propuesto se describen los siguientes pasos:
1. Un usuario quiere obtener un sello de tiempo para un documento electrónico que él posee.
2. Un resumen digital “técnicamente un hash” se genera para el documento en el ordenador del usuario.
3. Este resumen forma la solicitud que se envía al servidor de sellado de tiempo.
4. El servidor de sellado de tiempo genera un sello de tiempo con los datos del usuario, hash, la fecha y hora obtenida del servidor.
5. El sello de tiempo se guarda en la base de datos del servidor.
6. El servidor de sellado de tiempo mantiene un registro de los sellos registrados para su futura verificación.

Y para la verificación:
1. Se calcula el hash del documento del usuario.
2. El hash del documento del usuario es comparado con el hash registrado en la base de datos del servidor de sellado de tiempo para comprobar que son idénticos, evidenciando que el sellado de tiempo y el mensaje no fueron manipulados y que éste fue realmente registrado por el servidor. Si esto no ocurre, entonces el sellado de tiempo fue manipulado o bien el sellado de tiempo no fue emitido por el servidor de sellado de tiempo.

IMPLEMENTACIÓN DEL SIMPLE HASHING

La implementación del Simple Hashing se desarrolla usando los siguientes elementos:

Tabla 3. Lista de elementos a usar para la implementación del Simple Hashing

Elemento	*Categoría*	*Función*
Ubuntu 16.04 LTS	Sistema Operativo	Plataforma base
MySQL 5	Backend	Base de datos
PHP7	Backend	Motor dinámico web PHP
Apache2	Backend	Servidor HTTP
HTML5	Frontend	Lenguaje básico web
CSS3	Frontend	Lenguaje de diseño web
Javascript	Frontend	Lenguaje de contenido dinámico
jQuery	Frontend	Librería Javascript multiplataforma
Bootstrap	Frontend	Framework Javascript de interfaces
Datatables	Frontend	Librería Javascript para datos
Responsive	Frontend	Librería Javascript para contenidos
Select	Frontend	Librería Javascript de estilos de selección de datos

El esquema de trabajo de los elementos es el siguiente:

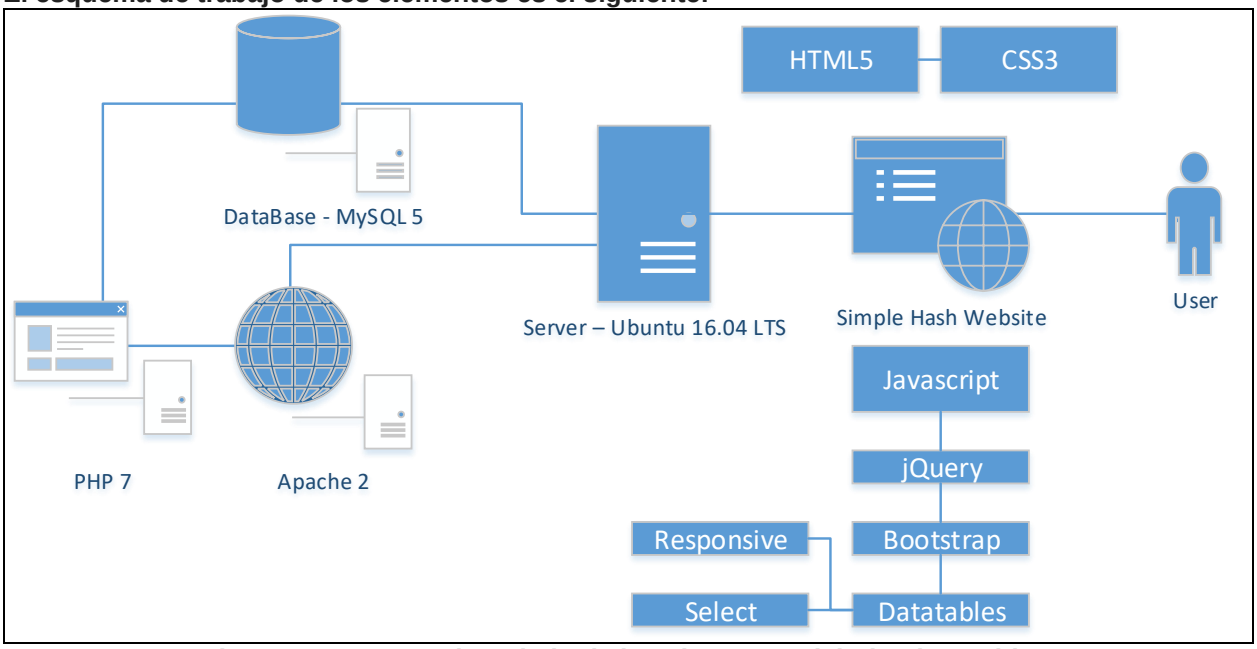


Figura 7. Esquema de trabajo de los elementos del Simple Hashing

Diseño de la base de datos

Para el diseño de la base de datos se han asignado campos que permitan realizar una auditoría. Es decir, para que sea posible cruzar información por email, fecha de creación, fecha de último ingreso, registros generados, registros borrados, registros inválidos, contraseñas duplicadas o débiles, etc.

En el campo de los hashes es necesario elegir un algoritmo de resumen de mensajes vigente, en este caso se hace uso de SHA256 y como hash secundario CRC32.

El diseño de la base de datos es el siguiente:

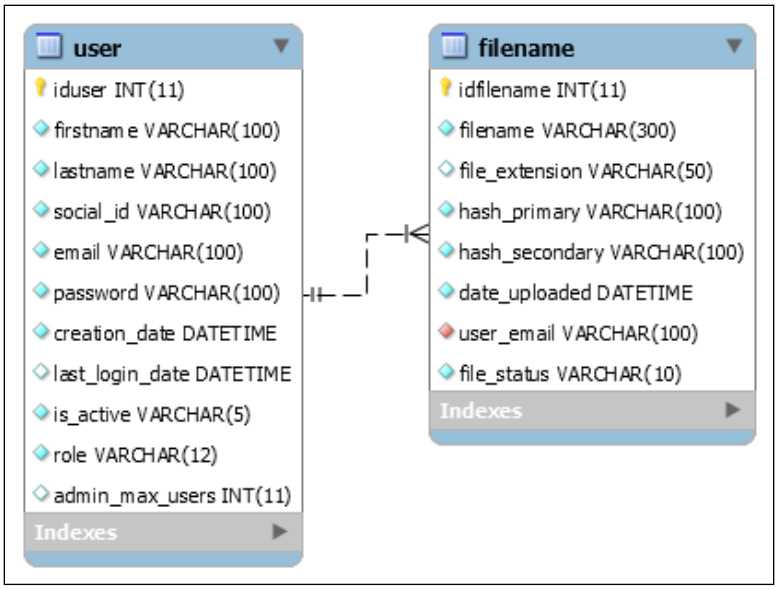


Figura 8. Diseño de la base de datos - Simple Hashing

Tabla 4. Descripción de los campos de las tablas de la base de datos – Simple Hashing

Tabla user		
Campo	**Descripción**	**Campo Auditable**
iduser	Llave primaria	No
firstname	Nombre	No
lastname	Apellido	No
social_id	Cédula	Sí
email	E-mail	Sí
password	Contraseña en SHA256	Sí
creation_date	Fecha de creación	Sí
last_login_date	Fecha de último ingreso	Sí
is_active	El usuario está habilitado	Sí
role	Rol del usuario	Sí
admin_max_users	Límite de usuarios que puede crear un administrador	Sí
Tabla filename		
idfilename	Llave primaria	No
filename	Nombre del archivo	No
file_extension	Extensión del archivo	No
hash_primary	Hash primario (SHA256)	Sí
hash_secondary	Hash secundario (CRC32)	Sí
date_uploaded	Fecha de registro	Sí
user_email	Email del usuario que registró el archivo	Sí
file_status	Estado del documento, (Válido, Borrado, Bloqueado)	Sí

Algoritmo – Inicio de Registro de Sellado de Tiempo

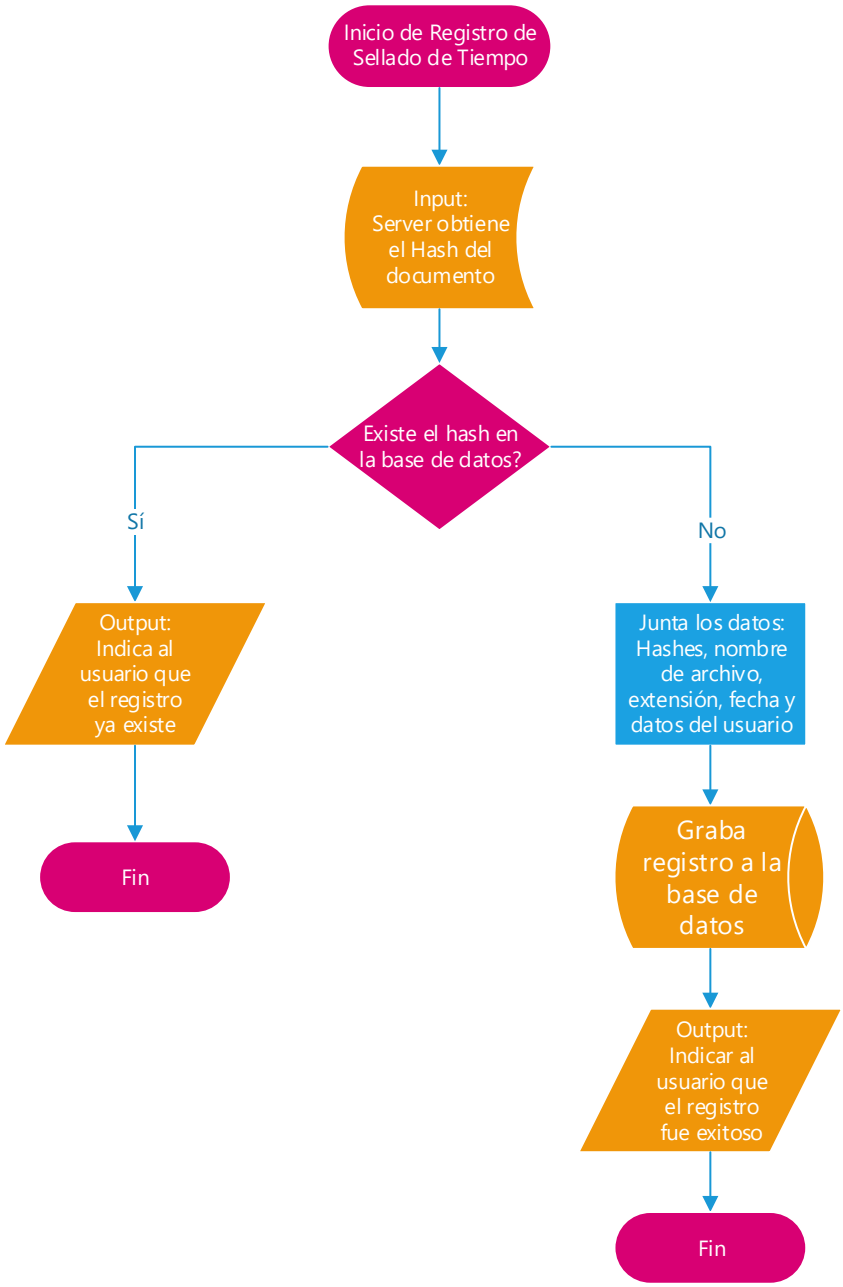


Figura 9. Inicio de registro de sellado de tiempo - Simple Hashing

Algoritmo – Verificación de Registro de Sellado de Tiempo

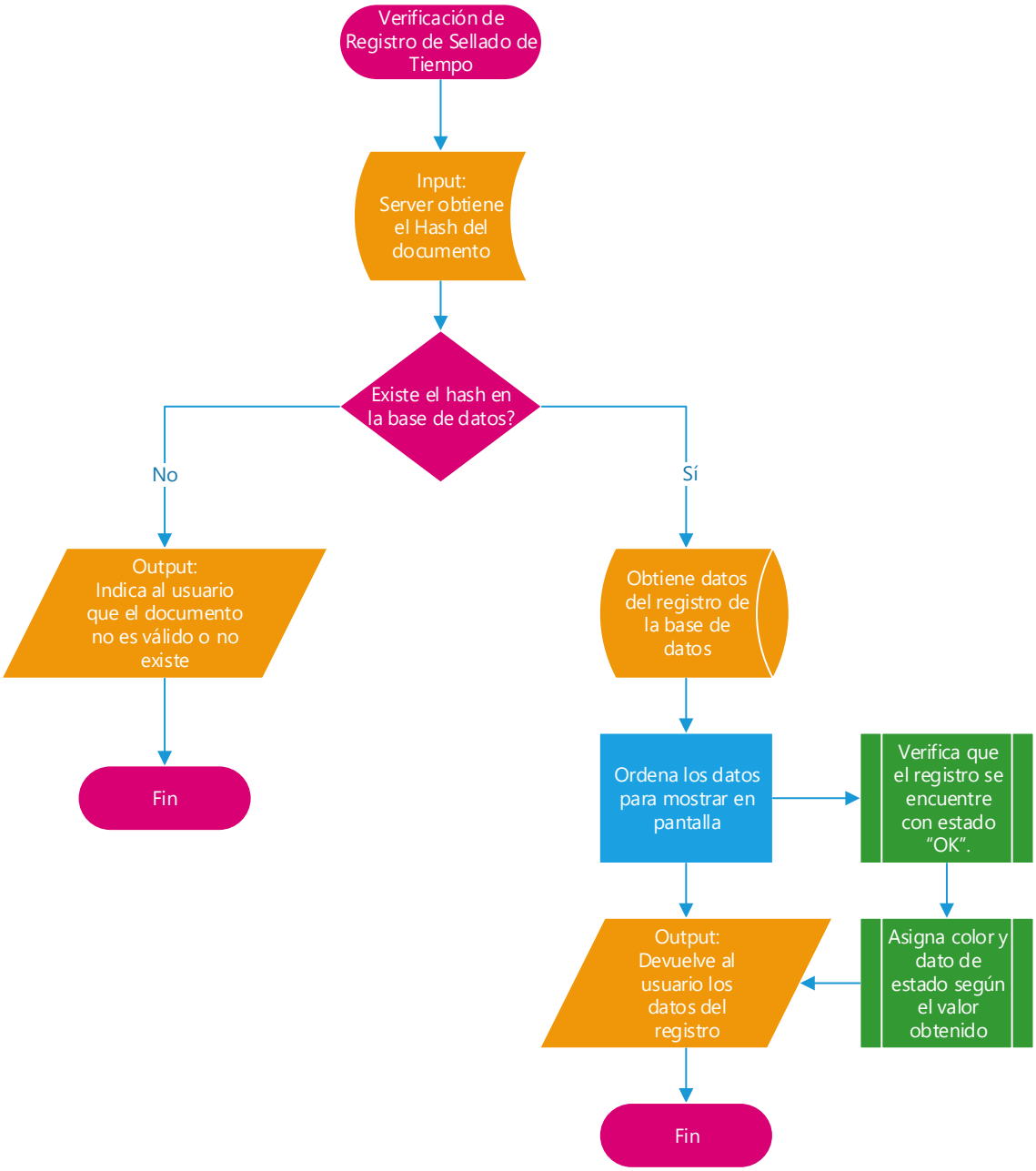


Figura 10. Verificación de registro de sellado de tiempo - Simple Hashing

Algoritmo – Cambio de estado del registro de Sellado de Tiempo

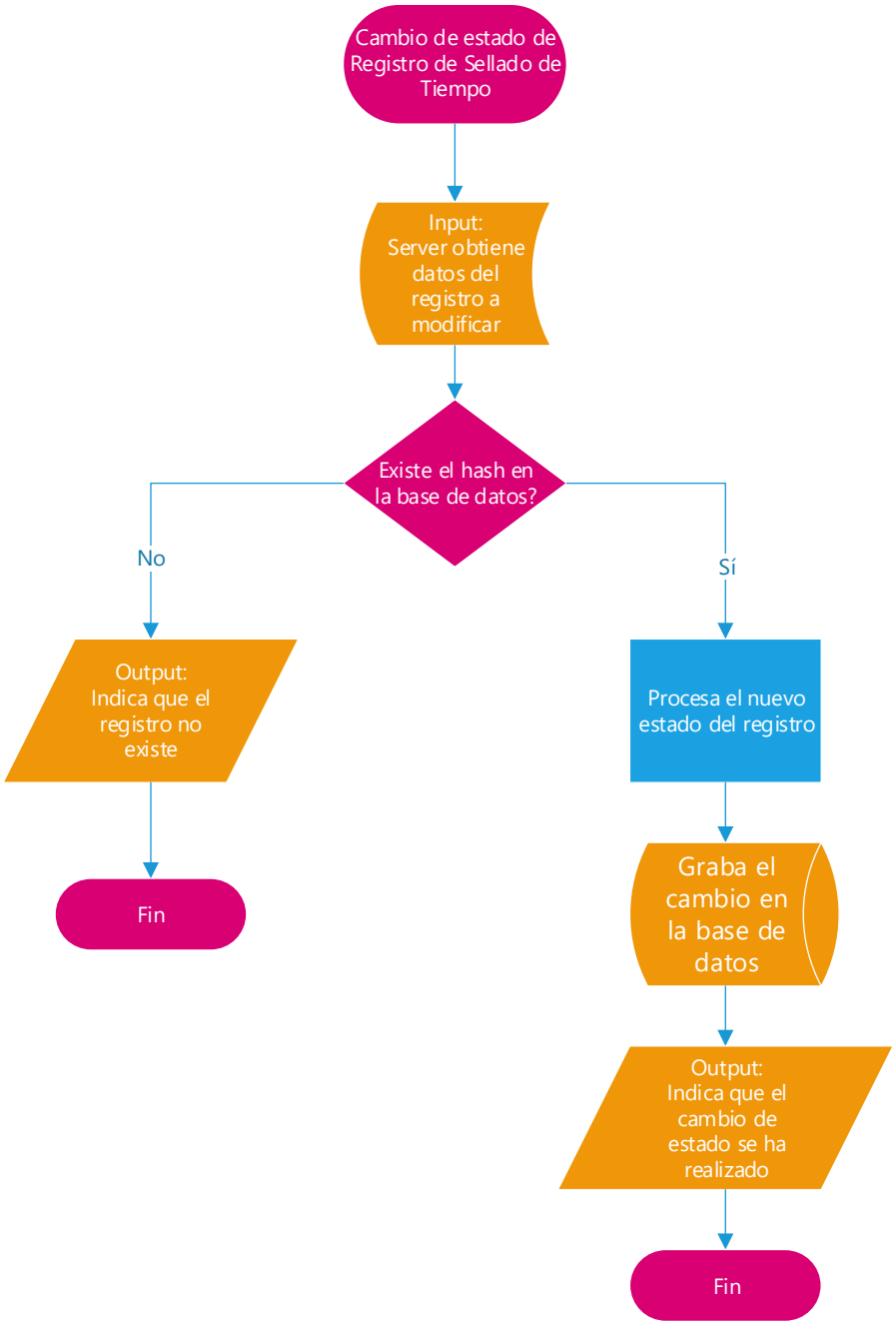


Figura 11. Cambio de estado de registro de sellado de tiempo - Simple Hashing

Algoritmo – Lógica
α = Hash del documento
β = Existe en Base de datos
ϰ = Graba Registro
δ = Muestra Mensaje
λ = Concatena datos
ε = Procesa estado de datos
str1 = Registro exitoso
str2 = El registro ya existe

str3 = El documento no es válido o no existe

Algoritmo de Registro

$\alpha(x) \wedge \neg\beta(x) \rightarrow \lambda(x) \wedge \lambda(x) \wedge \delta(str1) \ \therefore 1$
$\alpha(x) \wedge \beta(x) \rightarrow \ \delta(\mathrm{str2}) \ \therefore 0$

Algoritmo de Verificación

$\alpha(x) \wedge \neg\beta(x) \rightarrow \ \delta(\mathrm{str3}) \ \therefore 0$
$\alpha(x) \wedge \beta(x) \rightarrow \ \lambda(x) \wedge \varepsilon(\mathrm{x}) \wedge \delta(str[\varepsilon(\mathrm{x})]) \ \therefore 1$

Algoritmo de Cambio de Estado

$\alpha(x) \wedge \beta(x) \rightarrow \ \varepsilon(\mathrm{x}) \wedge \lambda(x) \wedge \delta(str1) \ \therefore 1$
$\alpha(x) \wedge \neg\beta(x) \rightarrow \delta(\mathrm{str3}) \ \therefore 0$

Es posible obtener los códigos fuentes en Github: https://github.com/RomelSan o pedirlos directamente al email: romel.vera.cadena@gmail.com

Se realizaron pruebas de funcionamiento a los algoritmos de registro, verificación y de cambio de estado con el fin de determinar la fiabilidad de los mismos.
Las pruebas se realizaron en una PC con Ubuntu 16.04 LTS y con los elementos adicionales que se mencionan en la tabla 3 que son requeridos para el correcto funcionamiento de la solución Simple Hashing.

Tabla 5. Prueba de Funcionamiento de Algoritmos

Tipo de Prueba	Cantidad de Pruebas	Pruebas Exitosas	Pruebas Fallidas
Algoritmo de Registro	20	20	0
Algoritmo de Verificación	50	50	0
Algoritmo de Cambio de Estado	9	9	0

En la tabla 5 se puede apreciar que todos los algoritmos funcionan correctamente. Para el algoritmo de registro se procedió a registrar 20 archivos aleatorios y se registraron con éxito. Para el algoritmo de verificación se procedió a usar 50 archivos de los cuales 20 se ingresaron previamente y devolvieron el hash correctamente. Los otros 30 archivos aleatorios se rechazaron puesto que no se encontraban registrados, por lo tanto, la prueba fue un éxito. Para el algoritmo de cambio de estado se procedió a marcar 9 archivos previamente ingresados como bloqueados, el cambio se reflejó exitosamente en la base de datos.

Tabla comparativa de soluciones similares

La información obtenida parte de los mismos datos que se encuentran en los sitios oficiales de las aplicaciones.

Tabla 6. Tabla comparativa de soluciones similares

Software	Chequea Integridad	Verifica Fecha	Depende de CA	SaaS	Comercial	Link
Digistamp	Sí	Sí	Sí	Sí	Sí	https://www.digistamp.com
SignFiles	Sí	Sí	Sí	Sí	Sí	https://www.signfiles.com
GlobalSign	Sí	Sí	Sí	Sí	Sí	https://www.globalsign.com
SignServer	Sí	Sí	Sí	Sí	Sí*	https://www.signserver.org
FreeTSA	Sí	Sí	Sí	Sí	Gratis	https://www.freetsa.org
QuickSFV	Sí	No	No	No	Gratis	http://www.quicksfv.org
Hashtab	Sí	No	No	No	Sí*	http://implbits.com

MultiHasher	Sí	No	No	No	Gratis	http://www.abelhadigital.com
Hashtools	Sí	No	No	No	Gratis	https://www.binaryfortress.com
Simple Hashing	Sí	Sí	No	Sí	Gratis	N/A

Nota: Sí* = ofrece una versión de costo de uso empresarial y una versión gratuita sin soporte y sin algunos componentes que se encuentran en la versión pagada.

Todas las aplicaciones en la tabla 6 tienen la capacidad de verificar la integridad de documentos informáticos, sin embargo, no todas dependen de un CA o verifican fechas y esto se debe porque cada aplicación está diseñada para resolver requerimientos diferentes.

CONCLUSIONES

Las organizaciones deben velar por la integridad de sus documentos y no necesariamente para lograrlo deben estar sujetas a un CA, especialmente cuando existen problemas de confianza. Por este motivo, existen varias soluciones que van desde una comprobación de integridad hasta operaciones más complejas en las que además de la integridad es posible también verificar otros parámetros como las fechas, firmas, origen de datos, autenticidad, etc.

La solución de sellado de tiempo "simple hashing" puede comprobar la integridad de archivos digitales y es posible emplearla como una solución alternativa cuando se requiera evitar problemas de confianza puesto que excluye a las autoridades certificadoras del esquema de sellado de tiempo. Adicionalmente, el algoritmo se puede modificar o adaptar para cualquier tipo de requerimiento, por ejemplo, la adición de parámetros de analíticas de google, revisión contra virus de los archivos en tiempo real, etc.

Posteriormente, es posible realizar un estudio de las implicaciones de seguridad (riesgos) del sellado de tiempo simple hashing en un ambiente funcional, siendo factible crear complementos o extensiones para soluciones existentes de manejo de contenidos, incluso se puede diseñar y analizar métricas de desempeño.

LIMITACIONES

El esquema Simple Hashing y la aplicación demo se desarrollaron en un entorno de laboratorio, es decir, en un ambiente de pruebas y de índole no empresarial, por lo tanto, debe considerarse como piloto, y de ser usado en un ambiente corporativo, se recomienda adaptar los parámetros y requerimientos de seguridad necesarios de la organización a la solución Simple Hashing.

BIBLIOGRAFÍA

Alrawi, O., & Mohaisen, A. (2016). Chains of Distrust: Towards Understanding Certificates Used for Signing Malicious Applications. *Proceedings of the 25th International Conference Companion on World Wide Web*, 451-456.

ANSI. (2016). Trusted Time Stamp Management and Security.

Apache. (2016). *HTTP Server Project*. Obtenido de http://httpd.apache.org/

Armstrong, R., & Mayo, J. (2009). Leveraging complexity in software for cybersecurity. *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*, 31.

Banco Central del Ecuador. (2011). Firma Electrónica. Obtenido de https://www.eci.bce.ec/marco-normativo

Barker, E. B., & Roginsky, A. L. (nov de 2015). *Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths.* National Institute of Standards and Technology (NIST). doi:10.6028/nist.sp.800-131ar1

Blaich, A. (2015). Decrease Your Circle of Trust: An Investigation of PKI CAs on Mobile Devices. *RSA Conference 2015*, (págs. 1-68). San Francisco.

Brenner, B. (30 de March de 2017). *Let's Encrypt issues certs to 'PayPal' phishing sites: how to protect yourself*. Obtenido de Naked Security by Sophos: https://nakedsecurity.sophos.com/2017/03/30/lets-encrypt-issues-certs-to-paypal-phishing-sites-how-to-protect-yourself/

Cassany, D. (2000). De lo analógico a lo digital. El futuro de la enseñanza de la composición. *Lectura y vida*, 6-15.

Corser, C. (2015). Google's Distrust of Chinese Digital Certificates. Obtenido de http://ncjolt.org/

Eckersley, P. (2015). *The Huge Web Security Loophole That Most People Don't Know About, And How It's Being Fixed*. Obtenido de Electronic Frontier Foundation: https://www.eff.org/about/staff/peter-eckersley

Flanagan, D. (2006). *JavaScript: The Definitive Guide.* OŔeilly Media. Obtenido de https://books.google.com.ec/books?id=2weL0iAfrEMC

Goodin, D. (2015). Stuxnet spawn infected Kaspersky using stolen Foxconn digital certificates.

Goodin, D. (2017). Google takes Symantec to the woodshed for mis-issuing 30,000 HTTPS certs. Obtenido de https://arstechnica.com/security/2017/03/google-takes-symantec-to-the-woodshed-for-mis-issuing-30000-https-certs/

Hars, A. (2002). Working for Free? Motivations for Participating in Open-Source Projects. *International Journal of Electronic Commerce, 6*, 25-39.

Johnson, C. S., Badger, M. L., Waltermire, D. A., Snyder, J., & Skorupka, C. (oct de 2016). *Guide to Cyber Threat Information Sharing.* National Institute of Standards and Technology (NIST). doi:10.6028/nist.sp.800-150

JQuery Foundation. (2016). *What is jQuery?* Obtenido de https://jquery.com/

Kaspersky, S. (2015). The Duqu 2.0 persistence module. Obtenido de https://securelist.com/the-duqu-2-0-persistence-module/70641/

Kim, D., Kwon, B. J., & Dumitraş, T. (2017). Certified Malware: Measuring Breaches of Trust in the Windows Code-Signing PKI. *CCS '17 Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (págs. 1435-1448). Dallas: ACM.

Kuhn, R., Hu, V., Polk, T., & Chang, S.-J. (2001). *Introduction to Public Key Technology and the Federal PKI Infraestructure.* NIST.

Lakhani, K. R. (2003). How open source software works: "free" user-to-user assistance. *Research Policy, 32*, 923-943.

Laurie, B. (2014). Certificate transparency. *Queue*, 10.

Lax, G., Buccafurri, F., & Caminiti, G. (2015). Digital document signing: Vulnerabilities and solutions. *Information Security Journal: A Global Perspective*.

MDN. (2016). *HTML developer guide*. Obtenido de https://developer.mozilla.org/en-US/docs/Web/Guide/HTML

OpenSource. (s.f.). *What is Linux?* Obtenido de https://opensource.com/resources/what-is-linux

Oppliger, R. (2014). Certification authorities under attack: A plea for certificate legitimation. *IEEE Internet Computing*, 40-47.

Paganini, P. (2014). *How Cybercrime Exploits Digital Certificates*. Obtenido de Infosec Institute: http://resources.infosecinstitute.com/cybercrime-exploits-digital-certificates/

Roosa, S. B., & Schultze, S. (2013). Trust darknet: Control and compromise in the internet's certificate authority model. *IEEE Internet Computing*, 18-25.

Shirey, R. (2007). Internet Security Glossary, Version 2. Obtenido de https://tools.ietf.org/html/rfc4949

Sklar, D. (2016). *Learning PHP: A Gentle Introduction to the Webś Most Popular Language.* OŔeilly Media. Obtenido de https://books.google.com.ec/books?id=-ZT4CwAAQBAJ

Specter, M. A. (2016). The economics of cryptographic trust: understanding certificate authorities. *Massachusetts Institute of Technology*.

Spurlock, J. (2013). *Bootstrap.* OŔeilly Media. Obtenido de https://books.google.com.ec/books?id=LZm7Cxgi3aQC

Stone, A. (2016). Protecting Data Means Balancing Security vs. Convenience. *GovTech Works*.

Tahaghoghi, S. M., & Williams, H. E. (2006). *Learning MySQL.* OReilly Media. Obtenido de https://books.google.com.ec/books?id=KQONxdHXS-wC

Trejo, C., Domenech, G., & Ortíz, K. (2016). LA SEGURIDAD JURÍDICA FRENTE A LOS DELITOS INFORMÁTICOS. *AVANCES*.

Turner, M., Budgen, D., & Brereton, P. (s.f.). Turning Software into a Service.

Une, M. (2001). The Security Evaluation of Time Stamping Schemes: The Present Situation and Studies. *IMES Discussion Papers Series 2001-E-18*, (págs. 100-8630).

Voutssas, M. (2010). Preservación documental digital y seguridad informática. *Investigación bibliotecológica*, 127-155.

W3C. (2015). *What is CSS?* Obtenido de https://www.w3.org/Style/CSS/

Zuccherato, R., Cain, P., Adams, D. C., & Pinkas, D. (2001). Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). RFC Editor. doi:10.17487/RFC3161

GLOSARIO

Código Abierto "Open Source"

El código abierto es el software desarrollado y distribuido libremente. Se focaliza más en los beneficios prácticos (acceso al código fuente) que en cuestiones éticas o de libertad que tanto se destacan en el software libre. Para muchos el término «libre» hace referencia al hecho de adquirir un software de manera gratuita, pero más que eso, la libertad se refiere al poder modificar la fuente del programa sin restricciones de licencia, ya que muchas empresas de software encierran su código, ocultándolo, y restringiéndose los derechos a sí misma. (Hars, 2002) (Lakhani, 2003)

Linux

Linux es un sistema operativo que sirve para que la computadora permita al usuario utilizar programas tales como: navegadores de internet, editores de texto, etc. Linux se origina a partir de Unix el cual apareció en los años sesenta y fue desarrollado por Dennis Ritchie y Ken Thompson del laboratorio telefónico de la empresa Bell. (OpenSource, n.d.)

HTML5

Es el lenguaje básico para la elaboración de páginas de internet, sus siglas HTML significan "HyperText Markup Language" en español "lenguaje de marcas de hipertexto" y el número cinco nos indica su revisión actual. (MDN, 2016)

CSS3

Es un lenguaje de diseño gráfico que permite definir y crear la presentación de un documento HTML, es muy usado para establecer el diseño visual de las páginas web o de cualquier otro lenguaje de marcado. (W3C, 2015)

Javascript

Javascript es un lenguaje de programación interpretado orientado a objetos. Se usa para mejorar la interfaz de usuario y para crear páginas web dinámicas del lado del cliente. Usa el dialecto ECMAScript estandarizado. (Flanagan, 2006)

Bootstrap

Bootstrap es un complemento de código abierto desarrollado por Twitter que tiene como objetivo facilitar el diseño web mediante un diseño adaptable, es decir que el diseño se ajusta a cualquier dispositivo y tamaño de pantalla. (Spurlock, 2013)

JQuery

JQuery es una librería multiplataforma de código abierto para javascript creada por John Resig en el año 2006, su objetivo principal es simplificar la interacción con los documentos HTML mediante la manipulación del árbol DOM, el manejo de eventos, la interacción con AJAX, etc. (JQuery Foundation, 2016)

Apache

Apache es un servidor web HTTP de código abierto creado por Robert McCool en 1995. Apache se usa principalmente para enviar páginas web estáticas o dinámicas por internet. (Apache, 2016)

PHP

PHP por sus siglas en inglés "Hypertext Preprocessor" es un lenguaje de programación creado por Rasmus Lerdorf en 1995 que funciona del lado del servidor y fue diseñado para la creación de contenidos web dinámicos. (Sklar, 2016)

MySQL

MySQL es un sistema de gestión de base de datos relacional de código abierto desarrollado por Oracle y se usa generalmente para entornos de desarrollo web. (Tahaghoghi & Williams, 2006)