CENTRO DE INVESTIGACIÓN Y DE
ESTUDIOS AVANZADOS
DEL INSTITUTO POLITÉCNICO NACIONAL

Unidad Zacatenco

Departamento de Computación

"Análisis y diseño de una aplicación segura de facturación electrónica para dispositivos móviles"

Tesis que presenta

Rodrigo Jurado Barrera

para obtener el Grado de

Maestro en Ciencias

en Computación

Directores de Tesis

Dr. Adriano de Luca Pennacchia

Dr. Moisés Salinas Rosales

México, D.F. **Febrero 2012**

Resumen

El uso de facturación electrónica en México se ha extendido en los últimos años desde su implantación en el 2004. El Sistema Tributario Mexicano (SAT) ha incluido ciertos mecanismos criptográficos para verificar la legitimidad de una factura electrónica. Sin embargo, el uso de estos mecanismos no resuelven todos los problemas de seguridad en el desarrollo de una sistema de facturación electrónica. El desarrollo de una aplicación segura, implica el uso de practicas, procesos, herramientas y técnicas que aborden los problemas de seguridad en cada fase del ciclo de vida de desarrollo de software.

En este trabajo se abordarán los problemas asociados a los riesgos de seguridad que pueda presentar una aplicación de facturación electrónica para ambientes móviles, y presentar así una propuesta para aminorar los riesgos de seguridad que conlleve desarrollar una aplicación de esta índole. También se propondrá el diseño e implementación segura de un esquema de facturación electrónica para generar facturas electrónicas para dispositivos móviles. Como parte del análisis se hace la descripción de los requerimientos de este tipo de aplicaciones, agregando a esta etapa procesos para la obtención y especifición de los requerimientos de seguridad. Una vez obtenidos estos requerimientos se realiza la descripción del diseño presentando una propuesta de una arquitectura que incluye el modelado de componentes de seguridad por medio de la herramienta UMLSec. Por último se muestra la implementación de la aplicación explicando la descripción de los componentes de seguridad que se desarrollaron.

Abstract

The use of electronic invoice in Mexico has spread in the last years since its introduction in 2004. The Mexican Tributary System (SAT) has included some cryptographic mechanism to verify the legitimacy of an electronic invoice. However, the use of these mechanisms do not solve all security issues in the development of an electronic invoice application. In order to make an application secure, the security analysis has to be an integral part of the system development process. The development of a secure application entails using practices, processes, tools, and techniques to address issues in every phase of the software development cycle.

In this work, we address the problems associated with security risks that may be present in an electronic invoice application for mobile environments, and show a proposal to lessen the security risks that entail the development of an application of this nature. Also it will be proposed the design and implementation of a secure electronic invoice scheme that generate electronic invoices for mobile devices. As part of the analysis, it is made a description of the requirements of these kind of applications adding to this phase, processes for eliciting and specifying the security requirements. Once obtained these requirements it is performed the design description which presents a proposal for an architecture that includes the modeling of security components through UMLsec.Finally, it is presented the implementation of the application stating the description of the security components that were developed.

Agradecimientos

Agradezco al CINVESTAV, ya que durante este tiempo curse materias que me apasionaron y pude ampliar mi percepción sobre las Ciencias de la Computación.

Agradezco al CONACYT, por el apoyo económico proporcionado a través de su programa de becas durante mi estancia en el Departamento de Computación del CINVESTAV.

Un sincero agradecimiento a mis directores de tesis, el Dr. Adriano de Luca Pennacchia y el Dr. Moisés Salinas Rosales por todo la ayuda y el apoyo que me fueron brindados durante la realización de este trabajo.

A Sofia Reza y a todo el personal secretarial por todo el apoyo brindado durante mi estancia en el Departamento de Computación.

Agradezco infinitamente a mi madre y abuelos por todo su apoyo, orientación, comprensión, amor y por estar en cada momento que los he necesitado. Sin olvidarme por supuesto de mi hermano y mi tio Octavio, quienes fueron parte importante en esta meta.

Agradezco a todos mis amigos de la maestría por su apoyo cuando este fue requerido, además por brindarme su compañia creando incontables momentos de diversión y varios buenos recuerdos.

A mis amigos y familiares por toda la confianza que depositaron en mi y me hayan alentado para llevar a cabo esta meta.

Por útimo, quiero hacer expresa mi gratitud de forma especial a todas las personas que jamás dudaron de mí, y que incondicionalmente brindaron su apoyo aún en las situaciones más difíciles.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Diseño de la investigación

Este capítulo se dedica a explicar el marco metodológico desde donde se aborda el contexto de este trabajo. Se hace una descripción del problema, el objetivo y la formulación general de la metodología que se llevará acabo para resolver el problema planteado.

1.1. Antecedentes

El gobierno electrónico o e-gobierno es un concepto que incluye el uso de tecnologías de la información para proporcionar a los ciudadanos facilidad en procesos de gestión pública y así aumentar la eficiencia de los servicios públicos reduciendo costos en consulta de información, trámites y servicios burocráticos.

En México a nivel Federal y Estatal se está participando activamente en importantes esfuerzos de iniciativas de e-gobierno para integrar soluciones de tecnología que ayuden a sustituir funciones tradicionales para así facilitar las operaciones de los gobiernos con el uso de tecnologías de la información. Como parte de estas iniciativas en el año 2004 el Servicio de Administración Tributaria (SAT) propuso un esquema de masificación para la emisión de Comprobantes Fiscales Digitales[2] (CFD) en su formato de factura electrónica que permite otorgar un medio de respaldo de las operaciones comerciales que realiza los contribuyentes y así remplazar las facturas de papel que se utilizan en la actual normativa. Contar con documentos digitales de este estilo tiene muchos beneficios entre los que se encuentran: agilizan la información contable, almacenamiento en formato digital, así como la reducción de costos y errores en el proceso de generación captura, entrega y almacenamiento.

Sin embargo debido a la aparición de este tipo de documentos digitales existe la obligación de replantearse muchas cuestiones de los documentos tradicionales en papel, surgiendo nuevos problemas, e incluso agudizando algunos de los ya existentes. En esta lista de problemas, se plantean cuestiones que van, desde la validez legal de los documentos digitales, la alteración de la información contenida en estos o que la

información sea comprometida. Debido a esto el SAT ha incorporado mecanismos para certificar e identificar de manera legítima el autor[2] de un CFD, a este mecanismo se le conoce como firma digital que es tratada tal y como si fuese la firma autógrafa. Sin embargo, la firma digital por sí misma es susceptible a varios ataques como lo son ataques de repetición, usurpación de identidad o ataques de hombre en el medio. Es por esta razón que el SAT incorpora el uso de estańdar X.509 de PKI (Infraestructura de Llave Pública [8]) el cuál nos ayuda a evitar los ataques anteriormente mencionados.

La infraestructura de llave pública (PKI) es un conjunto de hardware, software, políticas y procedimientos necesarios para crear, manejar, distribuir y revocar certificados digitales basados en criptografía asimétrica [27] . El principal objetivo por el cuál se desarrolló el concepto de PKI es para habilitar la adquisición de llaves públicas de manera segura, conveniente y eficiente.

A pesar de que el SAT incorporá el uso de estas herramientas criptográficas en el esquema de facturación electrónica, estás herramientas sólo resuelven ciertos problemas de seguridad. El diseño de un sistema que involucre los componentes de seguridad anteriormente mencionados aún incorporando estas herramientas puede ser susceptible a diversos ataques que comprometan la funcionalidad del mismo. El desarrollo de este tipo de sistemas de seguridad resulta ser una tarea difícil ya que muchos son diseñados e implementados sin satisfacer los requisitos de seguridad y esto trae como consecuencia ataques y vulnerabilidades en el sistema.

1.2. Problema y propuesta de solución

Las aplicaciones que prestan servicios a través de dispositivos móviles se han incrementado debido a la creciente proliferación de tecnologías de cómputo. Con la generalización de algunos servicios móviles, como el pago por móvil o la banca móvil es necesario proporcionar mecanismos que nos permitan proteger información crítica en este tipo de aplicaciones. El uso de aplicaciones móviles conlleva riesgos críticos de seguridad en la información la cuál es particularmente importante para los sistemas móviles, debido a la vulnerabilidad inherente de estos dispositivos. Para hacer frente a estos riesgos y permitir que la aplicación tenga el menor número de fallos de seguridad, analizar y modelar las amenazas potenciales que enfrenta una aplicación es un paso importante en el proceso de una aplicación segura. El proceso de análisis y modelado de aplicaciones trata de identificar cada uno de los problemas de seguridad relacionados al sistema: código, lógica del programa, flujo de información, datos, permisos de usuario y autenticación. Este proceso de análisis debe ser incorporado desde etapas tempranas de desarrollo e irse refinando en etapas posteriores. Por esta razón, tiene sentido identificar y documentar amenazas en tecnologías o aplicaciones específicas y así proporcionar directrices a proveedores de software de como mitigar los riesgos de seguridad.

Por otra parte con el advenimiento de servicios digitales se está tratando realizar la migración de documentos de carácter legal a su forma digital por esta razón existe la necesidad de implementar ciertas técnicas para garantizar la legalidad de estos y proporcionar la mismas características intrínsecas que su contraparte en papel.

Este trabajo de maestría abordará por una parte, los problemas asociados a los riesgos de seguridad que pueda presentar una aplicación de facturación electrónica para ambientes móviles, y presentar así una propuesta para aminorar los riesgos de seguridad que implica desarrollar una aplicación de esta índole. También se propondrá el desarrollo e implementación segura de un esquema de facturación electrónica para generar facturas electrónicas para dispositivos móviles.

1.3. Objetivos

General

Elaborar el análisis, diseño e implementación de un prototipo de un sistema que genere comprobantes fiscales electrónicos en su modalidad de factura electrónica aprovechando la tecnología de los dispositivos móviles para la automatización del proceso contable para el pago de impuestos de personas físicas y morales, utilizando metodologías de modelado seguro de aplicaciones.

Particulares

1. Realizar un análisis de requerimientos de seguridad utilizando la metodologiá SQUARE para una aplicación de facturación electrónica.

2. Realizar una propuesta basada en los requerimientos de seguridad que aminore los riesgos de seguridad en una aplicación de facturación electrónica en dispositivos móviles.

3. Diseñar una arquitectura robusta para ambientes móviles que realice facturación electrónica utilizando la herramienta de modelado UMLSec.

4. Implementar componentes para la arquitectura propuesta de facturación electrónica para ambientes móviles. La aplicación será capaz de generar comprobantes fiscales digitales en su modalidad de factura electrónica.

5. Realizar pruebas de seguridad y valorar resultados del producto obtenido.

1.4. Justificación

La masificación en el uso de télefonos móviles inteligentes y el aumento considerable en su poder de cómputo ha dado lugar al desarrollo de aplicaciones móviles avanzadas. Las aplicaciones que comÃºnmente son utilizadas en una computadora de escritorio están siendo migradas a sus versiones móviles. Estas aplicaciones van desde verificar el correo electrónico hasta aplicaciones más avanzadas de comercio electrónico.

Un problema grave dentro de estas aplicaciones móviles es que existe la posibilidad de poder acceder a datos confidenciales tales como información personal o datos de negocio. Es por está razón que para desarrollar aplicaciones móviles seguras es necesario el incorporar el uso de nuevas prácticas adoptando nuevas técnicas y metodologías en el ciclo de desarrollo convencional.

Por otra parte, cada aplicación móvil requiere distintos servicios de seguridad y tomando esto en cuenta resulta interesante incorporar el uso de metodologías de desarrollo seguro en el desarrollo de una aplicación de facturación electrónica, ya que puede estar expuesta a ataques como lo son: suplantación de identidad, elevación de privilegios, repudiación de la información, entre otros.

1.5. Metodología

Para facilitar la tarea del análisis y diseño de aplicaciones seguras existen metodologías que facilitan el modelado de componentes críticos y amenazas que puedan presentarse en un sistema de información con el fin de mitigar en medida de lo posible dichas amenazas.

Para el modelado de procesos de amenazas en el desarrollo de esta tesis se integrarán al ciclo de vida de convencional las siguientes fases de seguridad:

1. **Recolección de Requerimientos de Seguridad**: Se hará identificación de especificaciones formales de requerimientos y se hace una priorización de los objetivos de seguridad de la aplicación, valorando y clasificando riesgos utilzando la metodología SQUARE.

2. **Aseguramiento del Diseño**: Se especificará como será el manejo de las piezas de datos y de material criptográfico.A través de técnicas de modelado seguro como UMLSec donde se hará una especificación de diseño de los componentes de seguridad.

3. **Análisis a nivel implementación**: Se hará un revisión del código en busca de fallas de implementación, convenciones de seguridad y se hará uso de herramientas de análisis estático.

4. **Pruebas de seguridad**: Se harán pruebas específicas para encontrar efectos colaterales en la aplicación. Estas pruebas consisten en realizar pruebas funcionales y de penetración

1.6. Recursos

Para la realización de este proyecto se consideraron el uso de las siguientes herramientas:

1.6.1. Software

Sistema operativo Android

Android es un sistema operativo nacido de la alianza de 30 organizaciones del sector de los dispositivos móviles, como fabricantes de hardware, operadores y empresas de software, comprometidos a ofrecer un mejor teléfono móvil al mercado. El resultado es un sistema operativo y entorno de desarrollo de aplicaciones libre capaz de ejecutarse en distintos dispositivos. Android esta basado en el kernel de Linux y típicamente es programado con el lenguaje de programacón Java, lo que reduce la curva de aprendizaje y proporciona la facilidad y la seguridad del desarrollo de este lenguaje. Así mismo Android proporciona un conjunto de herramientas de abstracción variadas como lo son: interfaces de usuario, ciclo de vida de la aplicación, mecanismos de control entre procesos (IPC), y permisos.

Herramientas de desarrollo

Para la programación de aplicaciones sobre la plataforma Android es necesario el uso del SDK (Software Development Kit) de Android. El SDK ofrece herramientas para desarrollar y depurar aplicaciones incluyendo soporte para desarrollos en Linux, Windows y OSX. El SDK cuenta con un emulador bastante versátil que emula un dispositivo basado en ARM similar a un T-Mobile G1, aunque los parámetros de configuración de hardware virtual pueden ser modificados. El código que se desarrolla en la SDK se ejecuta sobre máquina virtual Dalvik y es posible depurar el código ya sea en el emulador o en un dispositivo físico. Para el desarrollo de aplicaciones sobre la plataforma Android puede ocuparse cualquier entorno de programación sin embargo para la realización de este proyecto y debido a su integración con el SDK de Android se ha optado por la utilización de Eclipse.

Software Adicional

Adicionalmente para el desarrollo de este trabajo se utilizaron las siguientes herramientas de software:

- Servidor Tomcat Apache 6.0
- Herramientas para análisis de código estático

1.6.2. Hardware

Para comprobar la funcionalidad de nuestro sistema se utilizó las siguientes terminales de cómputo:

Terminal Móvil

La terminal móvil utilizada como terminal de desarrollo fue un LG-OptimusOne (LG-p500h). El cuál tiene las siguiente caraterísticas:

- Sistema Operativo Android 2.2
- Procesador de 600 MHz
- 512 MB en RAM

Terminal Servidor

Para realizar ciertas funciones de sincronización de datos se utilizó una terminal con las siguientes características:

- Sistema Operativo GNU-Linux kernel 2.6
- Procesador Intel Core Duo de 2 GHz
- 2GB en RAM

1.7. Alcance

El sistema Generador de comprobantes fiscales digitales móvil está diseñado para ser utilizado por aquellos pequeños puntos de venta que no cuenten con la infraestructura para poder montar un sistema de facturación a gran escala. Por otra parte, el desarrollo del sistema será a través de software libre, se podrá contar con la memoria de diseño del software el cuál utilizó técnicas y herramientas de modelado seguro lo cual enriquece a la aplicación ya que aminorará los riesgos que se puedan presentar una aplicación de este tipo.

Capítulo 2

Facturación Electrónica en México

En este capítulo se abordarán las nociones básicas sobre las cuáles se fundamenta la facturación electrónica, así como las herramientas técnologicas que son utilizadas para garantizar la integridad, autencidad y no repudio de este tipo de documentos. Una vez introducidos estos conceptos se explicará como el Servicio de Administración Tributaria (SAT) implementa estos elementos en una plataforma tecnológica para automatizar el proceso contable para el pago de impuestos de personas físicas y morales en México.

2.1. Comprobantes Fiscales Digitales

Un Comprobante Fiscal Digital es un mecanismo alternativo de comprobación de ingresos, egresos y propiedad de mercancías en traslado por medios electrónicos. Maneja estándares de seguridad internacionalmente reconocidos, que garantizan que el comprobante es auténtico, íntegro, único y que es aceptado igual que el comprobante fiscal impreso [5]. Un Comprobante Fiscal Digital puede ser extensible a los siguientes tipos de comprobantes fiscales:

- Factura Electrónica
- Nota de Crédito
- Nota de Cargo
- Recibo de Honorarios
- Recibo de Arrendamiento
- Carta Porte
- Recibo de donativo

Para la realización de esta tesis se tomará como caso de estudio el uso de el comprobante fiscal digital en su formato factura electrónica. En la siguiente sección se explicará de manera breve el marco jurídico sobre el cuál se regula este tipo de documentos.

2.1.1. Definición Normativa

Una factura electrónica es la representación digital de un tipo de Comprobante Fiscal Digital (CFD), que está apegada a los estándares definidos por el SAT en el Anexo 20 de la Resolución de Miscelánea Fiscal, y la cual puede ser generada, transmitida y resguardada utilizando medios electrónicos. Cada factura electrónica emitida cuenta con un sello digital a través de la Firma Electrónica Avanzada (FIEL) que corrobora su origen y le da validez ante el SAT, una cadena original que funciona como un resumen del contenido de la factura, y un folio que indica el número de la transacción. La factura electrónica está regulada por los siguientes reglamentos y estatutos:

- Anexo 20 de la Miscelánea Fiscal, publicada en el Diario Oficial de la Federación junio 11, 2010.
- Art. 29 en el Código Fiscal de la Federación, publicado en el Diario Oficial de la Federación el 07 de diciembre de 2009.
- Resolución Miscelánea Fiscal del 2010.

2.1.2. Arquitectura de un Comprobante Fiscal Digital

Un comprobante fiscal digital debe de cumplir con los requisitos en el Código Fiscal de la Federación así como las reglas de la Resolución Miscelánea Fiscal y deberá de contar al menos con los siguientes datos:

- Datos del Emisor: razón social, domicilio fiscal y RFC (Registro Federal de Contribuyentes).
- Folio y en su caso serie.
- Lugar y fecha de expedición. La fecha deberá incluir la hora minuto y segundo en el siguiente formato: aaaa-mm-ddThh:mm:ss
- Datos del Receptor: RFC, en caso de ser un Comprobante Fiscal Digital global que amparen operaciones efectuadas de público en general se deberá utilizar el RFC genérico (XAXX010101000).
- Descripción de la mercancía y unidad de medida.

- Valor unitario, importe total y monto de los impuestos.
- La cadena original con la que se generó el sello digital.
- Sello digital correspondiente al comprobante fiscal digital.
- Número de serie del certificado de sello digital.
- Leyenda.
- Número y año de aprobación de los folios.

Cadena Original y Sello Digital: Se entiende como Cadena Original a la secuencia de datos formada con la información fiscal contenida dentro de cualquier Comprobante Fiscal Digital, la cual debe ser generada bajo las especificaciones establecidas por el SAT en el Rubro C de su Anexo 20. Se entiende como Sello Digital al conjunto de datos asociados al emisor y a los datos del documento por lo tanto es único e irrepetible por documento. Es el elemento de seguridad en una factura ya que a través de éste se puede detectar: si un mensaje ha sido alterado (integridad), determinar quién es el autor del documento (certidumbre de origen) y capacidad de impedir que el autor del sello digital niegue haber sellado el mensaje (no repudio). Ambos elementos deberán estar presentes en la representación impresa de la factura electrónica.

Leyenda: Todas las facturas electrónicas que sean impresas deberán portar la leyenda: "**Este documento es una impresón de un comprobante fiscal digital**".

En la figura 2.1 se muestra un ejemplo de una versión impresa de un Comprobante Fiscal Digital en su formato factura electrónica presentando sus componentes principales:

1. Información del cliente
2. Certificado, número de aprobación y año de aprobación
3. Serie, folio, fecha y hora de emisión
4. Sello digital
5. Cadena original
6. Leyenda

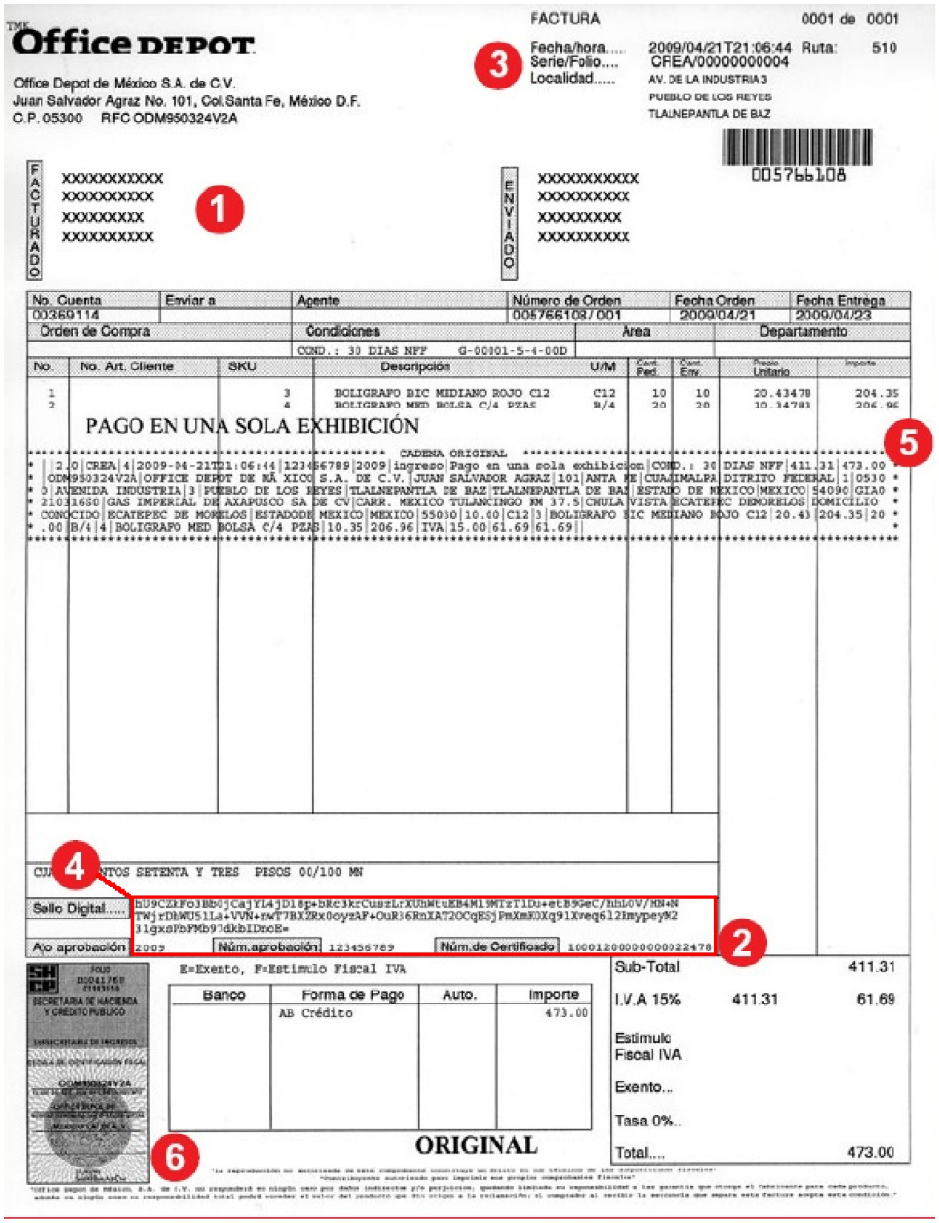


Figura 2.1: Elementos de un Comprobante Fiscal Digital

2.2. Herramientas Técnologicas asociadas a los Comprobantes Fiscales Digitales

2.2.1. Criptografía

La *criptografía* es el estudio de técnicas matemáticas relacionadas a aspectos de la seguridad de la información tales como la confidencialidad, la integridad de datos, la autenticación de entidad y la autenticación del origen de datos [21].

El objetivo principal de la criptografía es el de mandar datos de forma segura entre dos entidades através de un canal de comunicación, protegiendo así los datos intercambiados ante posibles ataques en nuestro canal de comunicación. Para satisfacer esta premisa, la criptografía ofrece los siguientes servicios de seguridad [21]:

- **Confidencialidad:** Es un servicio usado para mantener el contenido de la información secreto de todos exceptuando entidades autorizadas.
- **Integridad de datos:** Este servicio se encarga de prevenir la alteración no autorizada en los datos.
- **Autenticación:** Este servicio consiste en identificar las entidades que participan en un canal de comunicación. Adicionalmente este servicio provee la verficación de la información que es entregada sobre un canal autenticando su origen, fecha de origen, etc.
- **No repudio:** Mediante este servicio se garantiza que el emisor no pueda negar la creación de cierto mensaje o de las acciones que realizó.

Los métodos criptográficos están catalogados en dos, según la forma en que se manejan las llaves criptográficas: los de llave secreta o también llamados de llave privada y los de llave pública. Los métodos de llave privada son aquellos donde dos entidades tienen un método de cifrado/descifrado y ambos comparten una llave secreta. Por otra parte, los métodos de llave pública al igual que los de llave secreta comparten una llave secreta (comúnmente llamada llave privada), pero también cuentan con una llave pública.

2.2.2. Criptografiá de llave secreta

Como se había mencionado con anterioridad la criptografía de llave privada o simétrica es un método criptográfico donde se utiliza la misma llave tanto para cifrar y descifrar los mensajes. El esquema básico de método criptográfico se muestra en la figura 4.19.

Este sistema es simétrico con respecto a dos propiedades:

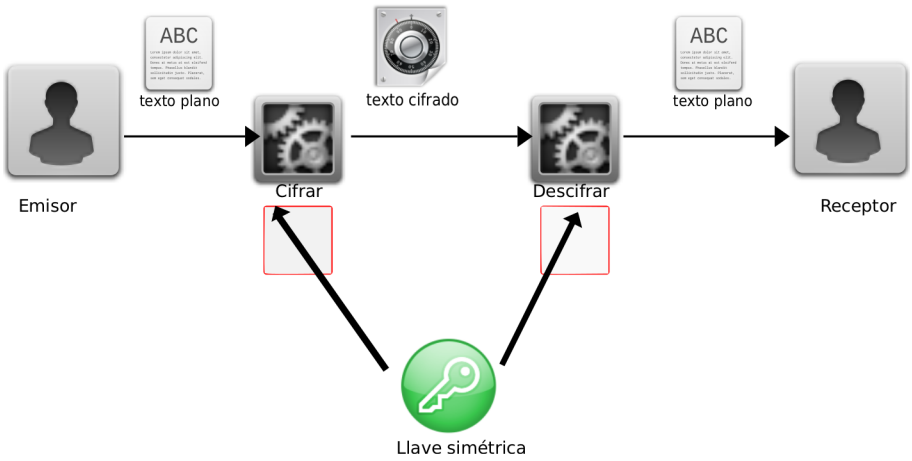


Figura 2.2: Esquema de criptogría de llave privada

- La misma llave secreta es usada para cifrar y descifrar.
- La *función* de cifrado y descifrado son muy similares.

Los algoritmos de cifrado simétricos modernos tales como AES (Advanced Encryption Standard) o 3DES (Triple Data Encryption Standard) son muy seguros, rápidos y usados ampliamente. Sin embargo, existen ciertas limitantes asociadas con los esquemas de cifrado simétrico, como los son[24]:

- **Problema de la distribuición de llaves**: La llave secreta debe de ser establecida entre las dos entidades en un canal de comunicación seguro. Cabe resaltar que enlace de comunicación para el envió de mensajes no es seguro, así que enviar la llave sobre este canal directamente es infactible.

- **Administración de el número de llaves**: Si cada par de usuarios necesitan un par de llaves en una red con n usuarios, entonces existen $\frac{n\cdot(n-1)}{2}$. Por ejemplo, en una corporación con 2000 personas, requiere más de 4 millones de pares de llaves que deben ser generadas y transportadas por medio de un canal seguro.

- **No existe protección contra fraude entre las entidades**: En el caso de algunas aplicaciones electrónicas es deseable probar que alguna de las entidades mando un mensaje. Sin embargo si se utiliza criptografía simétrica es posible que algunas de las entidades se retracte del envió del mensaje.

2.2.3. Criptografía de llave pública

Con el fin de tratar las limitantes que presenta la criptografía de llave secreta, Whitfield Diffie y Martin Hellman en el año 1976, en el artículo "New Directions in

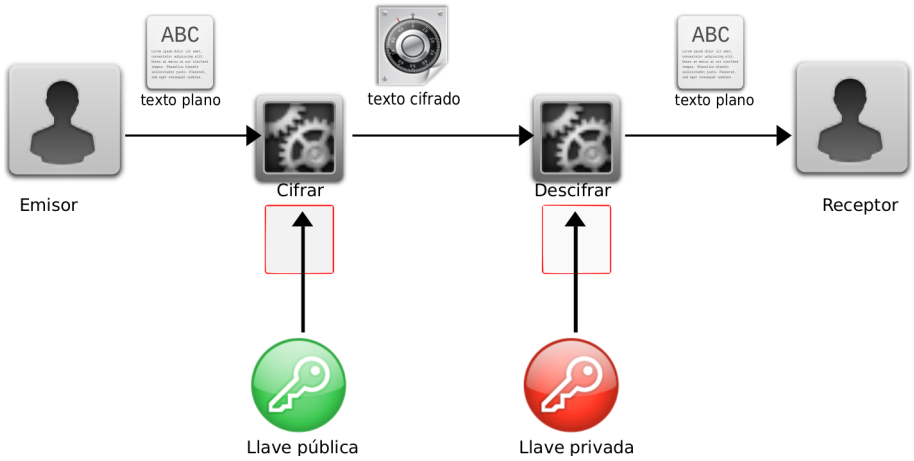


Figura 2.3: Esquema de criptogría de llave pública

Cryptography", hacen una nueva propuesta en la cuál, en vez de tener una sola llave secreta se contará con un par de llaves: una llave pública y una llave privada. La llave pública como su nombre lo indica, puede ser entregada a cualquier entidad que así lo solicite y es utilizada para cifrar la información por parte de alguna entidad emisora (si se trata de un esquema de firma electrónica esta utiliza una función de verificación por una entidad receptora). Por otra parte la llave privada es utilizada para descifrar el mensaje por alguna entidad receptora (en un esquema de firma electrónica esta se utiliza para realizar el algoritmo firma digital por alguna entidad emisora). En la figura 2.3 se muestra el esquema de criptografía de llave pública

2.2.4. Funciones Picadillo

Las funciones picadillo o funciones *hash* son utilizadas para garantizar la integridad de los datos. Una función picadillo es utilizada para construir una "huella digital"de una pieza de datos, si los datos son alterados, entonces la huella digital dejará de ser válida. Incluso si los datos son almacenados en un lugar inseguro, la integridad de los datos puede ser verificada de vez en cuando volviendo a calcular la huella digital y verificando que esta huella no haya cambiado. Una función picadillo tiene las siguientes características [31]:

1. La huella digital o digesto puede ser calculado de forma rápida.

2. Para piezas de datos de longitud arbitraria se producen huellas digitales de longitud fija.

3. Son funciones de solo ida. Una vez calculado la huella digital o digesto sobre una pieza de datos es imposible regresar al mensaje original.

4. Dado dos mensajes diferentes debe de ser computacionalmente imposible que tengan la misma huella digital (libre de colisiones). Esto es dado un mensaje x es imposible encontrar $x \neq x'$ tal que $h(x') = h(x)$.

2.2.5. Firma Electrónica

La firma electrónica es una de las herramientas criptográficas más importantes, su aplicación va desde los certificados digitales para establecer comercio electrónico seguro hasta el establecimiento de llaves sobre canales inseguros, constituyendo éstas, la instancia más importante para la criptografía de llave pública.

La propiedad de probar que una cierta persona generó un mensaje es muy importante fuera del dominio digital. En el mundo real esto se realiza através de las firmas autográfas en papel. Por ejemplo, si nosotros firmamos algún contrato o un cheque, el receptor puede verificar fácilmente que nosotros firmamos el mensaje. Al igual que con la firma autográfa convencional, solo la persona que cree un mensaje digital debe ser capaz de generar una firma válida. Las firmas digitales comparten algunas de las funcionalidades con las firmas autográfas. En particular, estas proveen un método para asegurar que un mensaje es auténtico a un usuario, es decir, que el mensaje se haya originado de la persona quien dice haber generado el mensaje. Para lograr esto es necesario el uso de la criptografía de llave pública y comunmente el uso de funciones picadillo. El uso de funciones picadillo nos ayuda a reducir el tamaño del mensaje manteniendo una longitud fija sin importar la longitud del mensaje original. Un esquema de firma digital esta compuesto de los siguientes componentes:

1. Un mensaje m.
2. Una función picadillo $h()$ la cuál debe de ser pública.
3. Un par de llaves: una llave privada k_{priv} y una llave pública k_{pub}.
4. Un algoritmo de firmado $sig_{k_{priv}}()$, así como su correspondiente algoritmo de verificación $ver_{k_{pub}}()$.

Los pasos para crear y verificar una firma digital son los siguientes:

1. Una entidad A calcula el picadillo de el mensaje $h_1 = h(m)$.
2. La entidad A firma el mensaje h_1 para obtener su firma $sig_{k_{priv}}(h_1)$.
3. El par $(m, sig_{k_{priv}}(h(m)))$ es enviado hacia la entidad B.
4. La entidad B aplica la función picadillo para obtener el digesto $h_2 = h(m)$.
5. La entidad B utiliza el algoritmo de verificación $h_3 = ver_{k_{pub}}(sig_{k_{priv}}(h(m)))$.

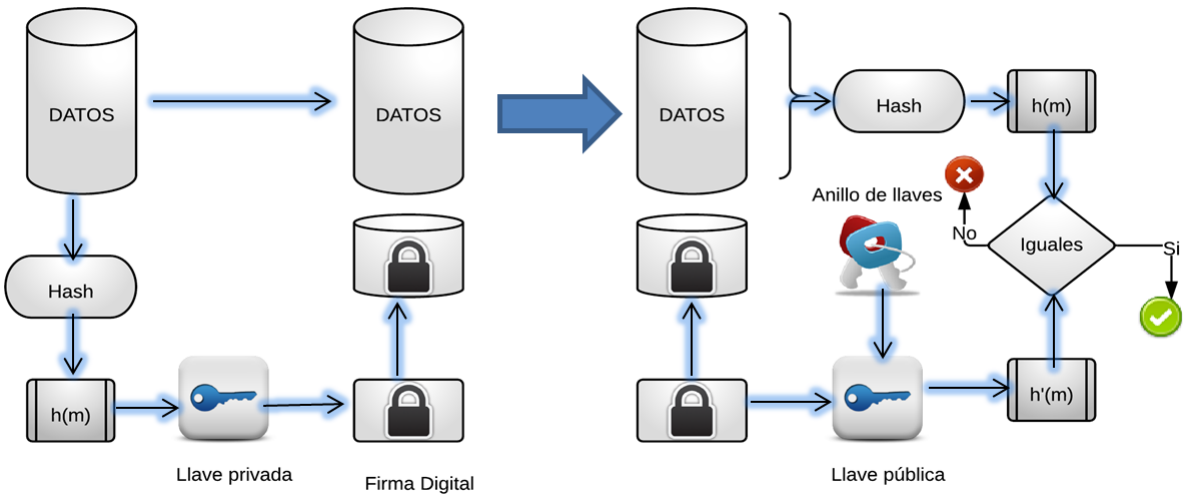


Figura 2.4: Esquema de firma digital

6. Si los digestos h_2 y h_3 son iguales esto quiere decir que la firma es autentica y el mensaje no sufrió ninguna alteración.

El proceso de firma digital puede visualizarse en la figura 2.4.

Con el uso de firmas digitales se pueden alcanzar los siguientes servicios de seguridad:

- Integridad
- Autenticación de Mensajes
- No repudio

2.2.6. Infraestructura de llave pública (PKI)

El uso de las firmas digitales por si solas es susceptible a diversos ataques como lo son ataques de repetición[0], usurpación de identidad[1] y ataques de hombre en el medio[2]. Es por esto que es necesario el uso de una infraestructura que evite este tipo de ataques.

[0]Ataques de repetición: Ataque donde un adversario repite o retarda la transmisión de datos de manera maliciosa.

[1]Ataques de usurpación de identidad: Ataque donde un adversario roba la identidad de un ente digital.

[2]Ataques de hombre en el medio: Este ataque esta relacionado al ataque de usurpación de identidad, donde un adeversario remplaza la llave(s) pública de alguno de los participantes de la comunicación por su propia llave.

La infraestructura de llave pública (PKI) es un conjunto de hardware, software, políticas y procedimientos necesarios para crear, manejar, distribuir y revocar certificados digitales basados en criptografía asimétrica [27] . El principal objetivo por el cuál se desarrolló el concepto de PKI es para habilitar la adquisición de llaves públicas de manera segura, conveniente y eficiente. En general la infraestructura de llave pública se compone de diversas tareas o actividades. Las tareas más importantes de la infraestructura de llave pública se mencionan a continuación [28]:

Emisión de certificados: Esta tarea se refiere a la emisión de nuevos certificados a los usuarios dentro de una PKI. La mayoría de las PKI's tienen una o más autoridades de confianza (usualmente llamadas autoridades de certificación). Antes que un certificado sea emitido, la identidad y las credenciales del usuario tienen que ser verficados por medios no criptograficos. Una vez verificados las credenciales del usuario, el certificado es emitido al usuario. Al mismo tiempo se realiza procedimiento criptográfico para generar la llava pública y privada para el propietario de el certificado. Una vez que se generó el certificado, este debe ser transmitido al propietario de una manera segura (posiblemente através de medios no criptográficos).

Revocación de certificados: Esta tarea se refiere a la revocación de certificados después de su fecha de expiración normal, la cuál debe de ser especificada en el certificado. La revocación de certificados puede darse por diferentes circunstancias imprevistas, tales como perdida o robo de la llave privada, o por uso fraudulento de la llave.

Respaldo de la llave/ recuperación/ actualización: El respaldar la llave se refiere a un almacenamiento seguro de las llaves privadas por el administrador de la PKI, en caso de perdida de las llaves privadas.

La recuperación de llaves es un protocolo que permite que una llave perdida sea restaurada o reactivada. Típicamente, un usuario tiene que probar su entidad antes de permitir el acceso a la llave privada.

Estampillado de tiempo: A la firma sobre cierto dato que incluya un cierto periodo específico tiempo durante el cuál la llave es válida se le llama estampillado de tiempo.

Comunicación segura: Este componente se refiere a protocolos seguros tales como el protocolo S/MIME (Secure Multipurpose Internet Mail Extensions), PGP(Pretty Good Privacy) o TLS(Transport Layer Security).

Control de acceso: El control de acceso también conocido como administración de privilegios incorpora autenticación, autorización y delegación. El control de acceso puede involucrar algunas formas de autenticación de usuario, ya sea por medio de una contraseña o algún esquema criptográfico de identificación.

Arquitectura de privacidad: Una arquitectura de privacidad permite el uso de certificados anónimos o pseudoanónimos. Este tipos de certificados podría mostrar la

membresía de un individuo en una clase de usuarios especificados, sin especificar su identidad particular.

2.2.7. Certificados Digitales

Los certificados digitales constituyen los bloques principales de la Infraestructura de Llave Pública y permiten un medio seguro y escalable en las PKI's. Un certificado digital vincula a una identidad con su llave pública. Esto usualmente sea hace por medio de el uso de autoridades certificadoras que firman la información del certificado, se supone que generalmente todos los usuarios tienen una copia válida de la llave púbilca de la autoridad certificadora. Por lo tanto, la firma de la autoridad certificadora puede ser verificada, lo cuál permite que la información en el certificado sea validada. Los certificados más comúnmente utilizados son los X.509 v3. Un certificado X.509 contiene los siguientes campos:

1. número de versión
2. número serial
3. ID del algoritmo de firma digital
4. nombre del emisor
5. periodo de válidez
6. nombre del propietario del certificado
7. la llave pública del propietario del certificado
8. campos opcionales
9. la firma digital de la autoridad certificadora sobre todos los campos anteriores

2.3. Plataforma tecnológica de la factura electrónica en México

La factura electrónica en México esta regulada por el Servicio de Administración Tributaria (SAT). A la representación electrónica de una comprobante fiscal impreso se le conoce como Comprobante Fiscal Digital(CFD), y que a partir de la nueva regulación será utilizado como comprobante de ingresos, egresos y propiedad de mercancías en México. Para la generación de Comprobantes Fiscales Digitales el SAT cuenta con ciertos mecanismos de seguridad, para garantizar que el comprobante sea auténtico, íntegro e único. Los componentes para poder generar Comprobante Fiscales Digitales son los siguientes:

- Firma Electrónica Avanzada (FIEL) y certificado de firma electrónica.
- Certificados de Sello Digital (CSD).
- Folios Digitales.

Los componentes anteriormente mencionados están unificados en una plataforma que utiliza los estándares de la Infraestructura de Llave Pública (PKI). En el resto de la sección se hablará de como se generan estos componentes y como es la creación de los comprobantes fiscales digitales.

2.3.1. Generación de la Firma Electrónica Avanzada (FIEL) y cerfificado de firma electrónica

Para la generación de la FIEL y el certificado FIEL, se deberán crear llaves de 1024 bits basados en el esquema de cifrado RSA. La llave pública deberá estar bajo el estándar *PKSC#10* y almacenada en un archivo de requerimientos con extensión `.req` en formato DER. Mientras que la llave privada se almacenará en archivo con extensión .key bajo el estándar *PKSC#8* en formato DER. Para la obtención de este par de llaves el SAT provee un programa llamado SOLCEDI(Solicitud de Certificados Digitales) basado en la biblioteca OPENSSL, sin embargo el contribuyente es libre de crear bajo sus propios medios este par de llaves bajo las especificaciones y los estándares anteriormente mencionados.

Por otra parte, para la obtención del certificado digital el contribuyente deberá realizar un cita ante las autoridades del SAT y presentar sus documentos de identificación (fotografía de frente, huellas dactilares, firma autógrafa,etc) junto con el archivo de requerimientos `.req` (almacenado en un medio digital), todo esto con el fin de vincular estos datos junto con la llave pública del contribuyente en un certificado digital. El SAT fungirá como autoridad certificadora y firmará el certificado con su llave privada. Al finalizar el trámite, se generará un certificado digital extensión `.cer` en formato X.509 v3 que se entrega en en el mismo medio digital en el cuál se llevo el archivo de requerimientos.

2.3.2. Certificados de Sello Digital (CSD)

Los certificados de Sello Digital (CSD) son un tipo de certificado digital que permiten acreditar la autoría de los Comprobantes Fiscales Digitales emitidos. Para obtener los Certificados de Sello Digital se requiere del Certificado de Firma Electrónica Avanzada (FIEL). A partir del programa SOLCEDI se deberá generar un archivo de requerimientos de certificado de sello digital que contendrá los datos del certificado de la FIEL y una llave privada para el uso del certificado de sello digital. Posteriormente el archivo de requerimientos de certificado de sello digital junto con la la llave privada

FIEL es ensobretado en un archivo .sdg siguiendo el estándar PCKS#7 para finalmente ser enviado vía un webservice a través de la aplicación CertiSAT. La aplicación CERTISAT será la encargada de generar el certificado de sello digital con extensión `.cer` y en formato X.509 v3.

2.3.3. Folios Digitales

Para poder iniciar a facturar electrónicamente es necesario solicitar los folios digitales. Los folios digitales constan de un número y una serie. Para obtener los folios digitales es necesario el uso de el programa SICOFI donde es necesario la autenticación del usuario ingresando la llave privada y el certificado de la Firma Electrónica Avanzanda. Posteriormente se deben de ingresar la llave privada y el certificado de sello digital y capturar la serie y el rango de folios que se requieran.

2.3.4. Modelo Operativo

En la figura 2.5 se muestra el modelo operativo entre el emisor y el receptor de Comprobantes Fiscales Digitales.

El modelo operativo para el envío y recepción de los comprobantes Fiscales Digitales es el siguiente: El emisor debe contar con sus folios, números de aprobación y su certificado digital correspondiente. Se genera un documento XML de acuerdo a como lo establece el Anexo 20. Este documento XML contendrá los datos necesarios de contabilidad de una factura electrónica, posteriormente se extrae la cadena original que no es más que una cadena que concatena varios campos del documento XML. La cadena original es firmado con el esquema de cifrado RSA utilizando el certificado. Al firmado de la cadena original se le conoce como sello digital. Para finalizar el proceso de generación de un comprobante fiscal válido es necesario agregar este sello digital al documento XML previamente generado. Una vez generado el Comprobante Fiscal Digital este debe ser almacenado por un periodo de 5 años por cualquier aclaración fiscal.

Al finalizar el proceso de generación del CFD al receptor se le hace llegar el documento de forma electrónica o magnética o impresa, para la integración de sus procesos de contabilidad, el cliente tiene la obligación de verificar los datos de la factura como lo son el Folio, Serie y Vigencia del Certificado.

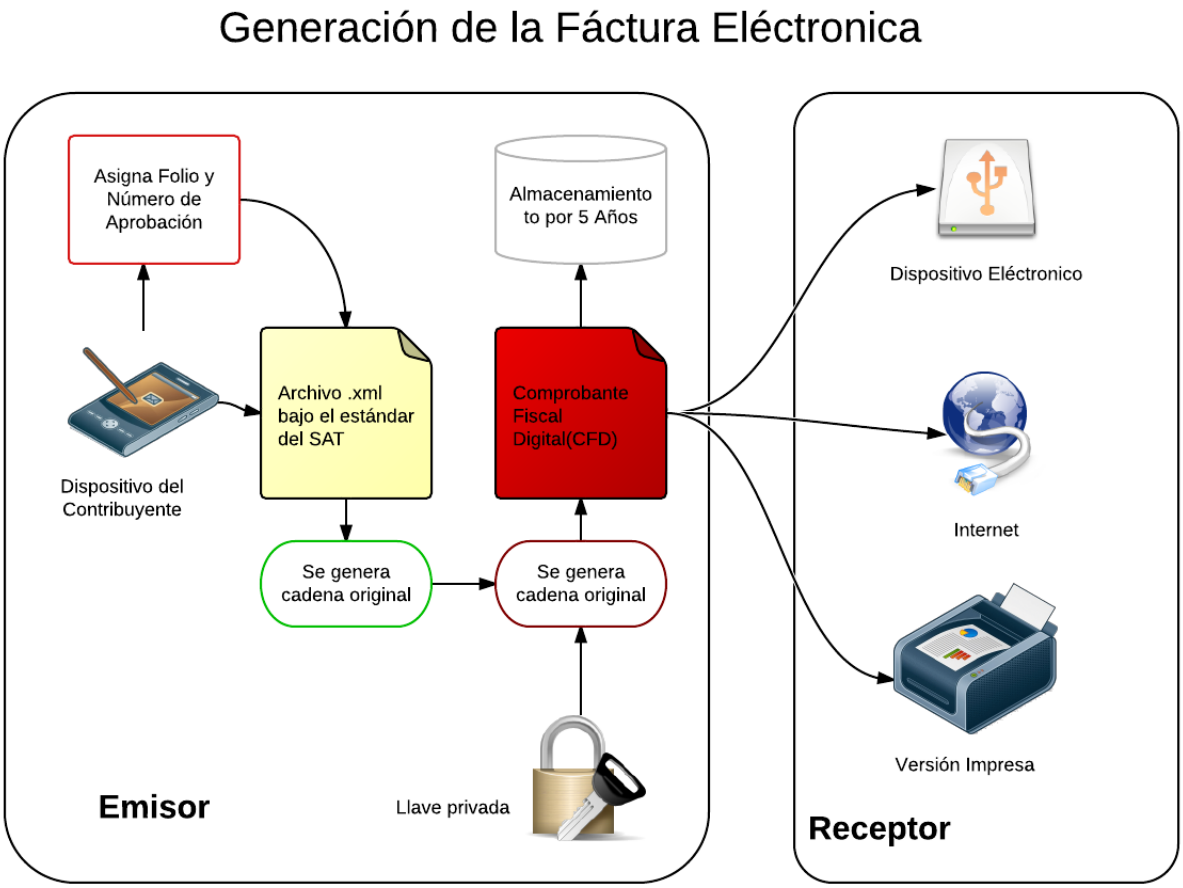


Figura 2.5: Modelo operativo de la Facturación Electrónica

Capítulo 3

Desarrollo de Software Seguro

En este capítulo se detallan los fundamentos teóricos que sirven como base para el desarrollo de aplicaciones de software seguro. Primeramente se introducirán de manera breve las metodologías clásicas de desarrollo de software para posteriormente hacer una descripción más detallada de las herramientas, técnicas y procesos clave para el desarrollo de software seguro haciendo énfasis en aquellos que fueron utilizados en el desarrollo de este trabajo.

3.1. Metodologías de Desarrollo de Software

Debido a la complejidad de las aplicaciones de software que se desarrollan hoy en día es necesario el uso de metodologías para facilitar y llevar de manera esquematizada el desarrollo de software. Las metodologías de desarrollo de software proporcionan un marco de trabajo para planear, estructurar y controlar el proceso de desarrollo de los sistemas de información. Existen diversas metodologías para el desarrollo de software a continuación se mencionan algunos de estos enfoques:

3.1.1. Modelo en cascada

El desarrollo aplicaciones de el modelo en cascada es un enfoque secuencial, en el cuál cada etapa del desarrollo de software debe de esperar la finalización de la etapa anterior, siguiendo rigurosamente las etapas de desarrollo análisis de requerimientos, diseño, implementación, pruebas, integración y mantenimiento.

3.1.2. Modelo en prototipado

Este modelo sugiere el desarrollo de software mediante la creación de prototipos, es decir la creación de versiones incompletas del software que se esta desarrollando.

La idea de este enfoque es perfeccionar el desarrollo de la aplicación y crear prototipos automatizados cada vez más sofisticados, poniéndolos a disposición de los usuarios para realizar pruebas hasta llegar así hasta un versión final del producto.

3.1.3. Modelo en espiral

El modelo en espiral es un proceso de software donde las etapas del desarrollo son colocados en una especie de espiral, donde cada iteración representa un conjunto de actividades. Este modelo de desarrollo combina las características del modelo de prototipado y el modelo en cascada.

3.1.4. Proceso de Unificado de Rational

El Proceso de Unificado de Rational(RUP)[26] es un ejemplo de un modelo de proceso que se apega a el uso de UML(Unified Modeling Language) y el asociado Proceso de Unificado de Desarrollo de Software. El Proceso de Unificado de Rational cuenta con las siguientes fases:

- *Inicio* El objetivo de la fase de inicio es el de establecer un caso de negocio para el sistema.
- *Elaboración* El objetivo es establecer un marco de trabajo arquitectónico para el sistema, desarrollar un plan de trabajo e identificar los riesgos que podrían surgir en el proyecto.
- *Construcción* Esta fase comprende el diseño del sistema, la programación y las pruebas. Al terminar esta fase, se debe de tener un software operativo y documentando.
- *Transición* Como su nombre lo indica esta fase busca hacer la transición de la comunidad de desarrollo hacia la comunidad de usuarios y hacerlo trabajar en un entorno real.

3.1.5. Desarrollo ágil de software

Este tipo de metodologías se basan en un enfoque incremental e iterativo, donde el cliente específica en cada incremento los requerimientos a incluir. Los clientes deben de estar fuertemente involucrados en todo el proceso de desarrollo, proporcionando y priorizando los nuevos requerimientos, evaluando cada iteración del ciclo de dasarrollo. En el desarrollo ágil de software se debe de optar por la simplicidad tanto en el software a desarrollar como en el proceso, trabajando de manera activa para elimnar la complejidad de el sistema.

3.1.6. Capability Maturity Model Integration

La integración de Modelos de Madurez y Capacidad (CMMI) es un proceso para la mejora y evaluación dentro de las organizaciones, provee a las organizaciones con los elementos esenciales para mejorar su desempeño[11]. CMMI puede usarse como una guía para la mejora ya sea en un proyecto, una división de la empresa o una organización entera. Por todas estas características CMMI es aplicable al desarrollo de software al uso de CMMI dentro de el desarrollo de software o servicios de información se le conoce como CMMI-DEV. CMMI-DEV es una colección de buenas prácticas que adoptan las organizaciones para mejorar la efectividad, eficiencia y la calidad de el software, también incluye guías que cubren los ciclos de vida del software desde su concepción hasta la entrega y mantenimiento del producto.

3.2. Desarrollo de Software Seguro

Dentro de los modelos y metodologías anteriormente mencionados, en ninguno de sus procesos se contempla la parte de desarrollo seguro de aplicaciones. Es por esta razón que para resolver este problema se ha desarrollado un proceso que facilite el modelado de componentes críticos y amenazas que puedan presentarse en un sistema de información. A este proceso se le conoce como ciclo de vida de desarrollo seguro el cuál busca mitigar en medida de lo posible ataques potenciales hacia el sistema de información.

3.2.1. Ciclo de vida de Software Seguro

El ciclo de vida de software seguro es un proceso que incluye una serie de procedimientos y actividades que deben de ser establecidas durante el ciclo de vida de desarrollo de software con el fin de desarrollar sistemas de información seguros y resistir en medida de lo posible los diferentes ataques que se puedan presentar en una aplicación. Reducir la debilidades de un sistema comienza desde la especificación de los requerimientos de seguridad. El software que incluye los requerimientos de seguridad puede ayudar a anticipar comportamiento anormal y nos ayudara a crear software confiable y seguro. Un proceso de ciclo de vida seguro debe de compensar las deficiencias en los requerimientos de software añadiendo prácticas basadas en riegos y comprueba la adecuación de las prácticas durante el ciclo de vida del software. A continuación se mencionaran dos aproximaciones de como puede ser llevado el ciclo de vida de desarrollo los cuáles integran fases o tareas adicionales al ciclo de vida convencional de desarrollo. Actualmente, se ha identificado dos propuestas las cuales cubren el proceso del ciclo de vida seguro.

3.2.2. Ciclo de vida de Desarrollo Confiable de Microsoft

El ciclo de vida de Desarrollo Confiable de Microsoft (o simplemente SDL) es un proceso que Microsoft ha adoptado para el desarrollo de software que necesita resistir ataques [14]. El proceso agrega una serie de actividades enfocadas a la seguridad y entregables en cada fase de el proceso de desarrollo de software. Estás actividades de seguridad y entregables incluyen la definición de requisitos de seguridad y actividades de evaluación durante la fase de requerimientos, modelo de amenazas para la identificación de riesgos durante la fase de diseño, el uso de herramientas de análisis estático de código y revisiones durante la implementación, además de pruebas orientadas y pruebas difusas durante la fase de pruebas. Durante la fase de verficación se realiza una revisión final del código y finalmente, durante la fase de lanzamiento, la revisión se lleva a cabo por el equipo de Seguridad de Microsoft, que es un conjunto de expertos que han sido asignados durante todo el ciclo de desarrollo. El ciclo de vida de Desarrollo Confiable de Microsoft esta compuesto de 12 fases como se muestra a continuación:

- *Fase 0: Educación y Conciencia:* Esta etapa se encarga de enseñar tanto al equipo de trabajo como a la organización de las amenazas que existen al desarrollar sistemas de información y las técnicas que existen para mitigar estas amenazas.

- *Fase 1: Iniciación del Proyecto:* Este fase consiste en los siguientes pasos:

  - Determinar si la aplicación es cubierta por el ciclo de desarrollo seguro (SDL).
  - Asignar un lider de seguridad.
  - Construir el equipo de seguridad.
  - Hacer una compilación de los bugs que van a ser mitigados.

- *Fase 2: Definir y seguir prácticas de mejores diseños:* En esta fase se asigna como mandato a los ingenieros encargados de proyecto a pensar en las características de seguridad y implementar diseños seguros, aplicando principios de diseño seguro.

- *Fase 3: Evaluación de riesgos del producto:* El propósito de la evaluación de riesgos es la de clarificar el nivel de esfuerzo para llevar a cabo los requerimientos del ciclo de vida de desarrollo seguro y así determinar como gastar los recursos al desarrollar el producto.

- *Fase 4: Análisis de Riesgo:* En esta etapa se realiza un modelado de amenazas el cuál es importante para construir software seguro ya que es la piedra angular para entender como el producto podría ser atacado y como podría ser defendido.

- *Fase 5: Crear documentos de seguridad, herramientas, y mejores prácticas para clientes:* Esta etapa es la encargada de la realización de documentos y guías que sean de ayuda para que los usuarios conozcan las implicaciones de seguridad y las configuraciones.

- *Fase 6: Políticas de código seguras:* Durante esta etapa se hace uso de las mejores prácticas de código seguro y se aplica el uso de herramientas de análisis de código seguro.

- *Fase 7: Políticas de prueba seguras:* En esta etapa se realizan diferentes tipos de métodos para hacer pruebas en la aplicación ya implementadas dentro de las que se encuentran: pruebas difusas, pruebas de penetración, verificación en tiempo de ejecución, revisar de nueva cuenta el modelo de amenazas, reevaluación de los ataques.

- *Fase 8: Push de seguridad:* Esta etapa consiste en hacer una revisión del código, incluyendo una actualización del modelos de amenazas y de la documentación de los ataques

- *Fase 9: Revisión de seguridad final:* Antes de que el producto pueda entregarse a los usuarios finales se debe de hacer una revisión final de seguridad verificando que todo el grupo de trabajo haya seguido correctamente el ciclo de desarrollo seguro.

- *Fase 10: Planificación de respuesta de seguridad:* Una vez que el software se encuentre listo es necesario crear un equipo o un centro de respuesta que este preparado para investigar y eliminar nuevas vulnerabilidades de seguridad.

- *Fase 11: Liberación del producto:* En esta fase se libera el producto después de haber pasado por la revisión de seguridad final.

- *Fase 12: Ejecución de respuesta de seguridad:* En esta fase se hace la ejecución de un plan de respuesta en caso de que aparezcan nuevas vulnerabilidades contando con toda la documentación que se realizó en las fases anteriores.

3.2.3. Ciclo de vida de seguridad mejorado de McGraw

El ciclo de vida de seguridad mejorado de McGraw y Taylor[30] compensa las deficiencias en los requerimientos de software, agrega prácticas de riesgo y verifica las adecuaciones de esas prácticas agregando el concepto de puntos clave de seguridad durante todas las fases del ciclo de vida de software. En la figura 3.1 se muestra como incorporar la seguridad en el ciclo de vida de desarrollo seguro. Los puntos claves de seguridad son aplicados a un conjunto de artefactos de software que son creados durante el proceso de desarrollo de software.

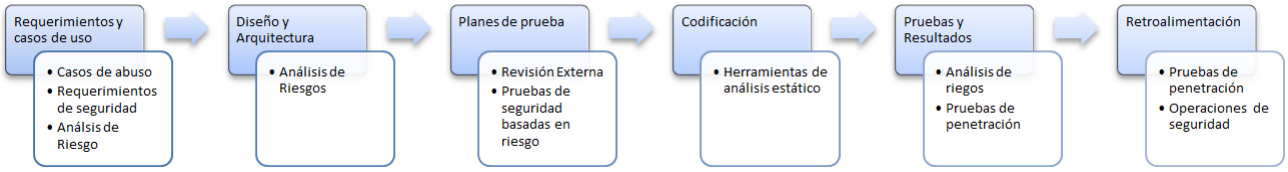


Figura 3.1: Ciclo de vida de seguridad de McGraw y Taylor

En la fase de seguridad de requerimientos se manifiesta la seguridad funcional así como las características emergentes de seguridad que un sistema debe de integrar. Una buena forma para cubrir los requerimientos de seguridad es através de los anticasos de uso. Los anticasos de uso describen el comportamiento del sistema cuando este se encuentra bajo un ataque, construirlos requiere abarcar las partes del sistema que deben de ser cubiertos, de quién y por cuanto tiempo.

En la fase de diseño y arquitectura, el sistema debe de ser coherente y presentar una arquitectura unificada de seguridad tomando en cuanta las reglas de seguridad. Los diseños, arquitecturas y el análisis debe de estar debidamente documentado identificando todos los posibles ataques. Tanto en la fase de arquitectura basada en especificaciones como en el diseño jerárquico de clases, el análisis de riesgos se necesita incorporar. Utilizar el análisis de riesgos nos ayuda a descubrir y clasificar los riesgos para así encontrar una forma de mitigarlos.

Para la fase de codificación trata de enfocarse en encontrar en las fallas de implementación utilizando herramientas de análisis estático (herramientas que exploran el código en busca de vulnerabilidades).

Para realizar pruebas de seguridad se deben de abarcar dos estrategias: la primera es probar la funcionalidad de seguridad con técnicas de pruebas funcionales y la otra es utilizar pruebas de seguridad basadas en el riesgo de ciertos patrones de ataque y en modelos de amenaza. Realizar pruebas de penetración también resultan útiles para ver los riesgos de la arquitectura probando el sistema en un ambiente real.

La última fase es el monitoreo de seguridad se hace una revisión continua checando el estado o el comportamiento de software durante el tiempo y aplicando un plan emergente a nuevas vulnerabilidades que aparezcan.

El enfoque de McGraw y Taylor presentado anteriormente es neutral y por lo tanto, puede ser usado con una gran cantidad de procesos de software(cascada, ágil, espiral, etc).

3.3. Herramientas de desarrollo seguro

Con el fin de facilitar la tarea del análisis y diseño de aplicaciones seguras durante el ciclo de vida de desarrollo seguro se han creado metodologías y herramientas

que ayudan a realizar el modelado de componentes críticos y amenazas que puedan presentarse en un sistema de información estás se utilizan con el fin de mitigar en medida de lo posible dichas amenazas. A continuación se mencionaran aquellas que son relevantes para este trabajo de tesis.

3.3.1. SQUARE

Es una métodologia que proporciona los medios para obtener, categorizar y priorizar los requisitos de seguridad para los sistemas de información y aplicaciones. El enfoque de esta metodología es la construcción de los conceptos de seguridad en las etapas tempranas del ciclo de vida de desarrollo. Este modelo también se puede utilizar para documentar y analizar los aspectos de seguridad de los sistemas para el manejo y envió de futuras mejoras y modificaciones a los sistemas. SQUARE esta compuesto de 9 pasos para la obtención de requisitos, los cuales se presentan a continuación:

1. **Consenso sobre los conceptos y su definición.** En esta estapa se realizan reuniones con los clientes para establecer un conjunto común de reglas de seguridad para establecer una base común de conocimiento.

2. **Identificación de los objetivos de seguridad.** Se realiza a nivel organizacional y es para establecer un conjunto de metas de seguridad priorizadas para el proyecto. Está etapa provee una verificación consistente con las políticas y el ambiente operativo de la organización.

3. **Desarrollo de los artefactos necesarios para definir los requisitos de seguridad.** Antes de que se realicen los requerimientos de seguridad es necesario contar con un conjunto de herramientas para hacer el levantamiento de requerimientos, estas herramientas van desde diagramas de arquitectura del sistema, diagramas de anticasos de uso, árboles de ataque, etc.

4. **Valoración de riesgos.** El objetivo de esta fase es la identificar vulnerabilidades y amenazas que pueda enfrentar el sistema, viendo la posibilidad de que estos se materialicen en un ataque y las consecuencias que este pueda traer.

5. **Selección de las técnicas para enunciación los requisitos.** En esta fase se elige una técnica de enunciación de requerimientos que se adapte tanto a los clientes como al equipo de desarrollo.

6. **Obtención de los requisitos.** En esta fase se enuncian los requerimientos de seguridad de acuerdo a la técnica utilizada. La mayoría de las técnicas proveen una guía detallada de como utilizar dicha enunciación.

7. **Clasificación de los requisitos.** Esta fase consiste en distinguir entre los requerimientos esenciales, metas y restricciones arquitecturales y clasificarlos de acuerdo a su relevancia.

8. **Priorización.** En esta fase se elige que requerimientos van a ser implementados de acuerdo a la clasificación de requerimientos que se realizo anteriormente.

9. **Revisión de los requisitos.** El objetivo de esta fase es revisar através de un método de inspección las ambiguedades, errores y incosistencias que se pudieran tener en los requerimientos.

3.3.2. STRIDE

Es un método de clasificación para categorizar amenazas conocidas de acuerdo con el tipo de vulnerabilidad que se utiliza por algún atacante. El término STRIDE es el acrónimo de "Spoofing identity, Tampering with data, Repudiation, Information disclosure, Denial of service, Elevation of privilege". Es decir, suplantación de identidad, manipulación de datos, repudio, revelación de información, denegación de servicio y elevación de privilegios.

- **Suplantación:** El ataque de suplantación se produce cuando un atacante usurpa la identidad de alguna otra persona obteniendo información de autenticación.

- **Modificación de datos:** Este ataque se produce cuando los datos que son transmitidos por un canal de comunicación o almacenados de manera persistente son alterados con propósitos malignos.

- **Repudiación:** Este ataque esta asociado con usuarios que niegan la autoría de una acción o evento en un sistema de información.

- **Revelación de la Información:** Este ataque está relacionado cuando el sistema o aplicación revela información a personas no autorizadas.

- **Denegación de Servicio:** Este ataque se refiere a la introducción de datos o información maliciosa para mantener algún sistema de información no disponible.

- **Elevación de privilegios:** Este tipo de amenaza se presenta cuando un usuario no autorizado obtiene privilegios que normalmente no tendría y debido a esto tiene el acceso suficiente para comprometer el sistema.

Para seguir el método STRIDE, se descompone el sistema en componentes significativos, tras analizar cada componente para comprobar si es susceptible de sufrir las amenazas anteriormente mencionadas y se proponen acciones que traten de mitigarlas. A continuación, se repite el proceso hasta llegar a una situación cómoda con las amenazas restantes.

3.3.3. UMLSec

Es una extensión de lenguaje de modelado UML para el desarrollo de sistemas seguros evaluando diagramas UML de varios tipos indicando sus vulnerabilidades. Necesidades periódicas de seguridad (tales como confidencialidad, integridad y autenticación) son ofrecidas como elementos de especificación por la extensión de UMLSec[16]. Estas propiedades son utilizadas para evaluar diagramas de varios tipos e indicar posibles vulnerabilidades.

Esta extensión esta dada en forma de un perfil de UML usando mecanismos de extensión de UML, utiliza estereotipos en conjunción de etiquetas de UML para formular requerimientos de seguridad y suposiciones del ambiente del sistema, así mismo usa restricciones que determinan si los requerimientos se cumplen con el diseño del sistema.

Para el desarrollo de sistemas críticos, este enfoque nos permite considerar los requerimientos de seguridad desde etapas tempranas de una manera transparente através del ciclo de desarrollo. Herramientas basadas en modelo como UMLSec nos servirán para establecer que el sistema cumplió con los requerimientos de seguridad a nivel diseño, analizando el modelo del sistema, y a su vez nos ayudara a verificar que el código que se realizó también es seguro generando secuencias de pruebas para el modelo.

3.3.4. Técnicas de Codificación Segura

Las prácticas de codificación segura ayudan substancialmente a reducir defectos comunes introducidos durante la implementación, por esta razón es importante conocer que tipos de agujeros de seguridad son comunes y presentar una estrategia de revisión a nivel código. Entre los hoyos de seguridad que podríamos encontrar durante la codificación encontramos los siguientes: validación incorrecta o incompleta de las entradas, manejo de excepciones pobre, buffer overflow, inyección de código SQL, condiciones de competencia, etc.

Para hacer las revisiones de código se deben de considerar estándares de codificación y hacer uso de reseñas de listas de verificación de código, inspeccionando los comentarios en código, documentación, pruebas unitarias y requerimientos de seguridad. En la fase de pruebas se detalla como el código debe usarse para demostrar que cumple con los requerimientos de seguridad y estándares de diseño/código destinados a reducir las fallas de diseño y errores de implementación.

3.3.5. Herramientas de Análisis de código estático

El análisis de código estático es el proceso en el cuál los desarrollores de software verifican su código para encontrar problemas y inconsistencias antes de compilar. Los desarrolladores pueden automatizar el análisis de código usando herramientas de análisis de código estático. Estas herramientas exploran el código fuente y detectan errores que típicamente el compilador no detectaría y las cuales podrían provocar problemas posteriores en el ciclo de vida de desarrollo.

3.3.6. Prácticas de codificación

Las prácticas de codificación típicamente describen métodos, técnicas, procesos, herramientas que pueden prevenir o limitar exploits hacia vulnerabilidades existentes. Las prácticas de codificación segura requieren el entendimiento de los errores de programación que suelen llevar a vulnerabilidades de software así como el conocimiento y uso de enfoques alternativos que son menos propensos a error.

3.4. Pruebas de seguridad de software

Las actividades de pruebas de seguridad son principalmente realizadas para demostrar que un sistema cumple con los requerimientos de seguridad, identificar y minimizar el numero de vulnerabilidades en el software antes de que el software sea lanzado a producción. El objetivo de las pruebas de seguridad es la de asegurar que el software que se este verificando sea robusto y continué su funcionamiento de una manera aceptable incluso en presencia de un ataque. Existen dos métodos para probar si un software ha cumplido con sus requerimientos de seguridad estas son las pruebas de seguridad funcionales y las pruebas de seguridad basadas en riesgos.

- **Pruebas Funcionales:** Estas pruebas tienen la intención de asegurar de que el software se comporte como se específico y se basa en demostrar que los requerimientos definidos se cumplan en un nivel aceptable.

- **Pruebas basadas en riesgo:** Las pruebas basadas en riesgo están dirigidas hacia los requerimientos negativos, los cuáles nos dicen lo que un sistema debe de hacer y lo que no. Estos requerimientos deben de venir de un análisis de riegos, el cuál debe debe de abarcar tanto los riesgos altos que existan como aquellos que no presentan tanto riesgo para el sistema de información.

El software es verificado en muchos niveles en un proceso típico de desarrollo. A continuación se mencionaran algunas actividades que son comunes en en la mayoría de los procesos de pruebas. alguno de ellos son repetidos en diferentes ocasiones para artefactos de software en diferentes niveles:

- **Pruebas Unitarias** Se utilizan usualmente en la primera etapa de pruebas e involucra probar relativamente pequeños componentes tales como clases, métodos, funciones, etc.

- Probar bibliotecas y archivos ejecutables

- **Pruebas de integración** El objetivo de esta prueba es la de verificar que cierta colección de componentes de software trabajen entre ellos correctamente.

- **Pruebas de sistema** Se pone a prueba todo el sistema, a través de pruebas de estrés o pruebas de penetración.

- **Pruebas de penetración** El objetivo de esta prueba es la de verificar que el sistema o aplicación sea comprometido a través de la búsqueda de vulnerabilidades.

Capítulo 4

Análisis y diseño de un sistema seguro de facturación electrónica

En este capítulo se presenta el análisis y diseño de un sistema de facturación electróncia explicando cuál fue el modelo de proceso a utilizar para la elaboración del sistema, incorporando a este proceso técnicas y metodologías de desarrollo de software seguro.

4.1. Análisis de Requerimientos

4.1.1. Perspectiva del producto

El sistema de facturación electrónica utilizará un dispositivo móvil para generar y certificar CFD's, contará con un mecanismo de sincronización para respaldar los CFD's. El envió de los CFD's serán enviados por correo electrónico a través de un canal seguro. La interacción con los usuarios será a través de una interfaz gráfica por medio de un dispositivo móvil.

4.1.2. Funciones del producto

El sistema de facturación electrónica tendrá las siguientes funciones:

- Crear factura
- Administrar clientes
- Crear factura
- Generar Reporte Mensual

- Reporte Contable
- Consulta de CFD's

4.1.3. Características del usuario

El sistema esta destinado a personas que quieran emitir facturas electrónicas en pequeños puntos de venta o pequeñas empresas a través de un dispositivo móvil.

4.1.4. Suposiciones y dependencias

El sistema tendrá las siguientes dependencias antes de poder usado:

1. La aplicación debe de estar firmada por el proveedor de el servicio de facturación electrónica.
2. La aplicación debe de tener acceso a la llave privada (`archivo .key`) previamente guardada en el dispositivo, esta debe ser obtenido a través del programa proporcionado por el SAT llamado SOLCEDI.
3. La aplicación móvil debe de tener acceso a el certificado digital(`archivo .cer`) de la FIEL el cuál contiene la llave pública del usuario. Este trámite se hace en las instalaciones del SAT.
4. La aplicación móvil debe de tener acceso a el certificado de sello digital (`archivo .cer`) generado por medio de la aplicación SOLCEDI y la aplicación CertiSAT. Este certificado sirve para firmar los comprobantes fiscales digitales.
5. El sistema debe de tener acceso a los folios electrónicos proporcionados por la aplicación SOLCEDI.

4.1.5. Requerimientos Funcionales

R.1 Facilidad de uso:

El sistema de facturación electrónica debe de contar con una interfaz amigable para el usuario.

R.2 Autenticación a través de usuario y contraseña:

Para ingresar al sistema de facturación electrónica se tendrá que ingresar mediante un nombre de usuario y contraseña, si no se cuenta con uno el sistema nos pedirá que ingresemos un nombre de usuario nuevo y contraseña.

R.3 Menú de Acceso:

Al entrar al sistema tendremos acceso a un menú que nos permitirá realizar las siguientes acciones:

- Crear factura
- Administrar clientes
- Crear factura
- Generar Reporte Mensual
- Reporte Contable
- Consulta de CFD's

R.4 Generar factura:

El usuario del sistema podrá realizar una factura electrónica seleccionando a un cliente de una lista de clientes registrados, en caso de que no exista el cliente el usuario podrá crear un nuevo cliente. El sistema deberá mostrar los datos del contribuyente a nombre del que se emite (receptor) la factura tales como:

- Nombre de la empresa.
- RFC.
- Domicilio Fiscal.
- Datos y Cantidad del producto.

Se requiere ingresar totalmente los datos para poder generar correctamente un CFD, de acuerdo a lo establecido en el anexo 20 sección C. Al tener los datos confirmados el sistema debe de crear la factura electrónica y debe de mandarse hacia receptor. Los CFD's emitidos se almacenan en el dispositivo en una colección de CFD's.

R.5 Crear Cliente:

El sistema tendrá la capacidad de ingresar un nuevo cliente. Para ingresar a un nuevo cliente el usuario tendrá que ingresar los siguientes datos:

1. **Razón Social**

2. **Correo electrónico**

3. **Dirección:**

4. **RFC**

Los datos del cliente se guardaran localmente en el dispositivo mediante una base de datos interna y se podrán respaldar en un servidor externo. Los clientes guardados por el sistema podrán ser utilizados para próximas veces en la creación de facturas.

R.6 Administrar clientes

El sistema podrá realizar cambios o bajas de clientes. El sistema debe de mostrar una lista con clientes teniendo la posibilidad de hacer bajas o modificaciones.

- **Modificación de Usuario** Al hacer una modificación se deben de mostrar toda la información del cliente(Razón Social, correo electrónico, dirección), el sistema tendrá la posibilidad de modificar esta información.

- **Baja de Usuario** Se tendrá que hacer una búsqueda del usuario del que se quiere que hacer la baja, a continuación se mostrará la información de dicho cliente, se contará con un botón para confirmar la baja de dicho cliente.

R.7 Generar Reporte Mensual

El sistema deberá generar el reporte mensual que debe de ser mandado al SAT. El usuario debe de seleccionar el mes y año del reporte. El reporte mensual debe de ser creado de acuerdo al Anexo 20 sección A, este reporte deberá ser guardado localmente en el dispositivo y posteriormente se debe mandar hacia el SAT por medio que este determine.

R.8 Reporte Contable

En el sistema se podrán visualizar todas las facturas que se hayan generado. El reporte contable deberá mostrar la siguiente información:

No. de Factura	Nombre	Importe
		Total

R.9 Consulta de CFD's

El sistema debe de contar con un mecanismo para buscar dentro de la colección de CFD's generados, el sistema podrá buscar el comprobante por los siguientes parámetros:

- Emisor
- Receptor
- Producto

Para ejecutar la consulta solo es necesario especificar alguno de los filtro especificados. Una vez localizado el/los CFD's acorde al criterio de búsqueda, se podrá consultar el detalle de cada CFD.

4.1.6. Requerimientos no funcionales

R.10 Mecanismo para almacenamiento seguro de los comprobantes fiscales

Los CFD's generados deben de ser almacenados en un deposito seguro en la memoria SD del dispositivo (copia local).

R.11 Mecanismo para respaldo de CFD's

Debe de existir un mecanismo para guardar de manera segura los CFD's en un repositorio remoto.

R.12 Mecanismo para sincronización de datos

El sistema debe de contar con un mecanismo de sincronización para respaldar los datos del usuario, clientes, CFD's asegurando que esta sincronización se haga a través de un canal seguro.

R.13 Envío de Comprobantes Fiscales Digitales

Una copia de los CFD's deben ser enviados al contribuyente a nombre que se emite (receptor) el cuál debe de verificar la validez de dicho comprobante, para lo cuál el cfd debe ser enviado por medio de un canal seguro. El envío puede hacerse a través de correo electrónico o hacia otro dispositivo por medio de bluetooth, por ejemplo.

R.14 Extracción de la información del Certificado Digital

Debido que el certificado digital generado por la aplicación SOLCEDI tiene la información relacionada con el emisor de los comprobantes fiscales digitales, se podrá extraer la información de el emisor para el uso del sistema de facturación electrónica, esta información debe de ser guardada localmente, garantizando el acceso exclusivo de la aplicación.

R.15 Sello Digital

El sello digital que se integra al CFD debe de hacerse de acuerdo al Anexo 20 sección D.

R.16 Lenguaje de programación

El lenguaje de programación será JAVA debido a que es el lenguaje nativo de la plataforma ANDROID.

4.2. Requerimientos de Seguridad con SQUARE

En esta sección se presentarán ejemplos de la aplicación de la metodología de SQUARE dentro de el contexto de una aplicación de facturación electrónica.

Debido a la naturaleza de este trabajo y tomando en cuenta que la aplicación de la metodología SQUARE se necesita un fuerte compromiso organizacional y debido a que ejecutar el proceso completo de la metodología es tardado, se ha decido el uso de una versión reducida de SQUARE llamada SQUARE-Lite.

SQUARE-Lite consiste en 4 pasos extraídos de la metodología SQUARE, estos pasos corresponden a los pasos 1,2,6,8. Los pasos en el proceso de SQUARE-Lite se encuntran resumidos en la siguiente tabla [10]:

Como puede observarse para algunos casos de la metodología es necesario trabajar directamente con el cliente para llegar a un consenso, la interacción con el cliente fue omitida haciendo suposiciones de las acciones organizacionales que debería de ejecutar el cliente.

4.2.1. Consenso sobre los conceptos y su definición.

Se definieron una lista de conceptos de seguridad que son indispensables para el desarrollo de la aplicación de facturación electrónica. Estas definiciones fueron tomadas de distintas referencias, como los son: el Anexo 20 del SAT, FIPS-140-3, RFC-2459, etc.Algunos de los conceptos que se definieron fueron los siguientes:

Paso	Entrada	Técnicas	Participantes	Salida
1 Acuerdo en las definiciones	Definiciones candidatas de estándares	Entrevistas estructuradas	Clientes, equipo de requerimientos.	Consenso de las definiciones.
2 Identificar metas de seguridad	Definiciones, metas candidatas, políticas.	Sesiones de trabajo, encuestas, entrevistas	Clientes, ingeniero de requerimientos.	Metas
3 Obtención de requerimientos de seguridad	Artefactos, resultado del análisis de riesgos, selección de técnicas	Modelo basados en análisis, lista de requerimientos, entrevistas, encuestas, método acelerado de obtención de requerimientos	Ingeniero de requerimientos.	Corte inicial en los requerimientos de seguridad
4 Priorizar requerimientos	Requerimientos categorizados, resultado del análisis de riesgos	Métodos de priorización	Ingeniero de requerimientos.	Requerimientos priorizados

Cuadro 4.1: Metodología SQUARE-Lite

- **Confidencialidad:** Es la propiedad de que la información sensitiva no este disponible o se divulge a individuos, entidades o procesos no autorizados [23].
- **Llave privada:** Llave criptográfica, usada con un algoritmo criptográfico, que esta únicamente asociado con una entidad y esta no se hace pública [23].
- **CFD:** Es un mecanismo alternativo de comprobación de ingresos, egresos y propiedad de mercancías en traslado por medios electrónicos. Maneja estándares de seguridad internacionalmente reconocidos, que garantizan que el comprobante es auténtico, íntegro, único[5].

Una lista completa de las definiciones utilizadas en este trabajo podrá encontrarse en el Apéndice A.

4.2.2. Identificación de metas de seguridad

Para nuestra aplicación de facturación electrónica se definieron políticas de seguridad. Recordemos que estas políticas se deben de realizar a nivel organizacional verificando que concuerde los objetivos de negocio de la organización. A continuación

mostraremos un ejemplo de objetivo de seguridad identificado para el contexto de la aplicación a desarrollar:

Objetivo: Integridad de datos. Los datos como la llave privada, folios, certificado digital, cartera de clientes, factura emitida y entregada (por entregar), copia local de facturas emitidas, copia de respaldo de facturas emitidas, reporte mensual no deben de ser alterados. Antes de utilizar el sistema de facturación electrónica se debe de realizar una función de autoprueba que cheque la integridad tanto de la aplicación como de los datos anteriormente mencionados, si la aplicación ha sufrido alguna alteración en los datos el sistema tendrá la capacidad de alertar al usuario.

Una lista de las metas de seguridad que fueron definidas podrán encontrarse en el Apéndice B.

4.2.3. Obtención y priorización de los requerimientos de seguridad

Casos de uso

Para definir el funcionamiento de nuestra aplicación y las interacciones que iba a tener el usuario con el sistema, fue necesario utilizar herramientas tales como los casos de uso. Un caso de uso es una técnica que se basa en escenarios para la obtención de requerimientos identificando una lista de interacciones particulares entre el usuario y el sistema. Un caso de uso comprende:

- El usuario con el que interactua el sistema (actor).
- Una descripción de la meta a alcanzar.
- Suposiciones que deben conocer para para que se realice el caso de uso con éxito.
- Una lista de los pasos que se realizan entre el actor y el sistema.
- Variaciones o caminos alternativos para realizar la meta.

Los casos de uso especifican la variedad de formas para utilizar el sistema. Estos definen las funcionalidades requeridas por el sistema. En nuestro caso los casos de uso fueron de gran ayuda para identificar servicios de misión crítica y los activos que pueden estar involucrados en el sistema.

Los casos de uso desarrollados en este trabajo se realizaron en base al modelo de negocios de descrito en el Cápitulo 2. Un ejemplo de caso de uso desarrollado se muestra en la tabla 4.4. Este caso de uso detalla las acciones del usuario con el sistema para generar una factura electrónica, además contiene excepciones de las acciones en caso de ocurrir algún funcionamiento anormal en el uso de la aplicación. Una lista completa de todos los casos de uso desarrollados en este trabajo podrá encontrarse en el Apéndice C.

Identificador	Generació de factura	
Nombre	Acción “Generar Factura”	
Descripción	El usuario podrán generar una factura electrónica a un cliente	
Precondición	El usuario debe de estar previamente registrado en el sistema.	
Actores principales	Usuario.	
Secuencia normal	Paso	Acción
	1	El usuario selecciona la opción “Crear Factura” dentro el menú principal del sistema.
	2	El usuario selecciona el cliente de una lista de clientes registrados(contribuyente), en caso contrario se podrá ingresar un nuevo usuario.
	3	Al seleccionar un cliente el sistema, mostrará los datos del cliente tales como: Nombre de la empresa, RFC, Domicilio Fiscal, Datos y Cantidad del producto.
	4	El usuario debe de de insertar los datos del producto(s) por el cuál se desea realizar la factura. Se deben de ingresar todos los datos del producto para poder realizar la factura. Los datos que se deben de insertar son los siguientes: Cantidad, Descripción, Si genera IVA, Precio Unitario. Este proceso se realiza por cada producto o servicio del cuál se requiera la factura.
	4.a	El sistema calcula los siguientes datos: Importe, Subtotal, Precio con IVA, Retención de IVA(Si aplica), Retención de ISR(Si aplica), TOTAL
	5	Una vez insertados todos los datos se muestran para confirmación y se puede proceder a realizar la factura electrónica, seleccionando el botón crear factura.
	6	Al momento de presionar el botón, se realizaran los siguientes pasos:
	6a	Se genera un archivo XML, con los parámetros que apliquen para dicha factura. Estos parámetros serán los que que se especifican en el Anexo 20 sección C.

Secuencia normal	Paso	Acción
	6b	Se procede a generar el sello digital de acuerdo al Anexo 20 sección D. Para realizar este proceso es necesario cargar la llave privada en la memoria , al finalizar el sello digital, la llave privada debe de ser borrada de la memoria de manera segura del dispositivo. Los pasos a seguir para realizar el sello digital son los siguientes: 1) Se genera la cadena original, 2) Se recupera la llave privada desde el repositorio seguro, 3) Con la llave privada creada en memoria se realiza la firma digital de la cadena original, 4) Se borra de memoria la llave privada, 5) Se anexa al documento xml el sello digital.Si existe algún problema con la llave o la firma digital se procederá a la excepción **signerr.**
	6c	Una vez generado el archivo XML y el sello digital, se procederá a enviar este archivo vía un canal seguro hacia el cliente(contribuyente).Se guarda las facturas en el repositorio seguro.
	6d	Si se generó y se envió correctamente el archivo XML, el sistema mostrara una notificación de éxito En caso contrario se procederá a la excepción **generr** o **generr2** según sea el caso**.**
	7	Se muestra la opción de regresar al menú principal.
Postcondición	La facturas son guardadas en formato XML(solo se mantendrán en este formato y no se tendrá una base de datos relacional para su almacenamiento).	
Excepciones	Excepción signerr	
	Paso	Acción
	1	Si en el momento de hacer la firma, la llave privada no se encuentra dentro de la memoria SD o si existe un error de entrada/salida al leer la llave privada se realizan los siguientes pasos:
	1.a	Si el sistema notifica un error al no encontrar la llave privada en la memoria SD el sistema notificará al usuario y se cerrará la aplicación
	1.b	Si el sistema notifica un error de entrada/salida al leer la llave privada el sistema notifica al usuario y cerrara la aplicación.

	Excepción generr	
	Paso	Acción
	1	La generación del archivo tuvo un problema al momento de escribirse a disco o no se realiza correctamente el sello digital.
	2	El archivo XML no se manda correctamente al contribuyente.
	3	Se lanza una notificación al usuario para que vuelva a realizar la factura
	Excepción generr2	
	1	Existe un problema de conexión y no se puede enviar el archivo XML
	2	El sistema guarda el archivo y lo marca como pendiente para poder mandarlo posteriormente y recuperarse de dicha excepción.
Frecuencia esperada	Cada vez que el usuario genere una factura	
Importancia	Alta, indispensable para la generación de comprobantes fiscales digitales.	

Cuadro 4.4: Generación de fa

Como se había mencionado con anterioridad los casos de uso fueron de gran importancia ya que nos ayudaron a identificar los activos relevantes para nuestra aplicación. Un activo es un elemento que se busca proteger ante las posibles vulnerabilidades del sistema. Dentro de los activos identificados en este trabajo tenemos los siguientes:

- llave privada
- folios
- certificado digital
- cartera de clientes
- factura emitida y por entregar
- copia local de facturas emitidas y estado
- copia respaldo de facturas emitidas
- dispositivo

Casos de Abuso

Para diseñar un software confiable y seguro, es necesario anticipar comportamiento anormal en el sistema. Los casos de abuso nos pueden ayudar a ver el software que se este diseñando de la misma forma que un atacante lo haría. Pensando más allá de las funcionalidades convencionales y contemplando eventos inesperados o negativos en el sistema.

Los casos de abuso de uso son similares a los casos de uso tradicionales, excepto que estos están hechos para detallar un abuso, amenaza o vulnerabilidad de el sistema y como este podría solucionarse.

Los casos de abuso nos fueron de gran ayuda en el diseño del sistema de facturación electrónica particularmente en la obtención de los requerimientos de seguridad, además de que nos ayudo a reconocer los ataques a los que podría estar expuesta nuestra aplicación proporcionando un conjunto de recomendaciones para mitigar esas vulnerabilidades.

Un ejemplo de caso de abuso desarrollado en este trabajo se muestra a continuación:

Numero:	MC-01	
Nombre:	Genera Factura sin acceso a llave privada(residual)	
Alcance:	Compromete la misión de la aplicación	
Prioridad:	Prioridad Alta	
Ambiente:	Host y Aplicación.	
Anti-Actores:	Usuarios no Autorizados	
Niveles de derecho de acceso:	Nivel Usuario Nivel Administrador	
Puntos de entrada:	Aplicación	
Atributos de seguridad afectados:	Autenticidad de la información,no repudio	
Descripción:	Una vez que se ha generado una factura previa, es posible que la llave privada haya quedado en memoria. Dicha copia podría usarse para generar una factura posteriormente y sin que esto requiera verificar al contenedor de la llave privada.	
Precondiciones:	Se accede a la llave privada para firmar una factura electrónica de forma local en el dispositivo(se requiere que por lo menos se haya generado una firma para poder realizar el ataque). Se accede al proceso gen_factura() y se mantiene en memoria la copia de la llave privada. No se inicializó la variable que contiene la llave privada. No se cerro la sesión de la applicación. Se tenian folios cargados en el dispositivo.	
Suposiciones:	Se supone que este ataque se realiza mediante el robo del dispositivo.	
Post-Condiciones:	Peor Caso	Se genera fáctura válida sin aviso al contribuyente
	Medida de prevención deseada	-Blanquear llave privada -Lock de la aplicación.
	Medida de detección deseada	-Bitacora de la aplicación realizando un corte diario -Cruce de las facturas emitidas contra el SAT.
	Medidia de recuperación deseada.	-Revocar la llave privada. -Litigio.
Perfiles de potenciales anti-actores:	Usuarios sin conocimiento técnico agudo.	
Clientes y amenazas:	Clientes: creación de facturas sin consentimiento del contribuyente.	
Amenazas relacionadas:	Elevación de privilegios, acceso no autorizado a la interfaz,autenticidad de los datos, usurpación de la identidad,repudiación de la información.	
Recomendaciones arquitecturales:	-Se debe de realizar un bloqueo de la aplicación cada 10 min si la aplicación no esta en uso.-Cada vez que se realice un factura se de pedir el password de la aplicación o la del acceso a la llave privada.-Crear bitacoras de las operaciones dela aplicación.-Blanqueo de la llave privada cada vez que se ocupe.	
Políticas de Recomendación:	-Realizar cortes de caja al día y ver las facturas que se han emitido-El password que se utilice para generar la factura electrónica debe de ser cambiado periódicamente.-Contar con políticas para contraseñas fuertes.-Acciones de usuarios deben de ser revisadas periódicamente.	

Cuadro 4.5: Caso del Mal Uso MC-01

Como se puede apreciar los casos de abuso casi contiene los mismos parámetros que contendría un caso de uso convencional, sin embargo también conceptualiza las medidas de seguridad que deberían tomarse si este ataque fuera exitoso, incluyendo las prevenciones que deberían de realizarse pare prevenir, detectar y recuperarse de ese ataque. Otro aspecto a resaltar en la realización de los casos de abuso es que en cada caso de abuso se agrego el apartado Amanezas relacionadas. Este apartado nos sirvió para ver las amenazas al que el sistema esta expuesto en caso de que el caso de abuso se realice con éxito, esta clasificación se realizó de acuerdo al sistema de clasificación STRIDE. Una lista completa de todos los casos de uso desarrollados en este trabajo podrá encontrarse en el Apéndice D.

Árboles de ataque

Un árbol de ataque muestra una vista potencial de forma jerárquica de como se realiza un ataque en un sistema, basado en como los ataques pueden realizarse. A continuación se muestra el siguiente árbol de ataque para el caso de la aplicación de factura electrónica.

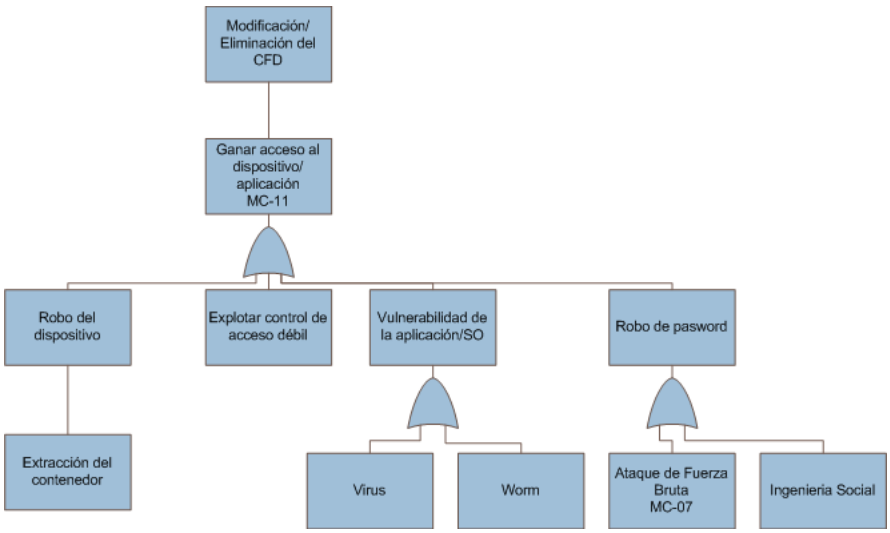


Figura 4.1: Modelo operativo de la Facturación Electrónica

En la figura 4.1 se muestra un subárbol de ataque que corresponde a la acción malintencionada de una modificación o eliminación de una factura electrónica. En el ejemplo se supone que un adversario ha obtenido acceso al dispositivo donde se emite la factura. El nodo raíz representa el objetivo del adversario a realizar para nuestro caso se denotó bajo la leyenda **ganar acceso al dispositivo**. Cada nodo no raíz representa un submeta, y los hijos de ese nodo son formas de realizar dicha meta. Se puede notar que los árboles de ataque existen nodos con compuertas OR o AND. Todos los nodos compuertas OR representan alternativas para cumplir el objetivo de el nodo padre. Por ejemplo en la figura para realizar robo de password este se podría

obtener por ataque de fuerza bruta o por ingeniería social. Un árbol también puede contener nodos con compuertas AND las cuales representan los diferentes pasos que se deben para que el adversario realice el objetivo del nodo padre. Una vista completa de los arboles de ataque desarrollados en este trabajo podrá encontrarse en el apéndice C.

Estimación de riesgos

Una vez obtenidos los activos y las amenazas a las que podría estar expuesto el sistema através del uso de herramientas como lo fueron casos de uso, casos de uso y arboles de ataque, es necesario realizar una estimación del estado de riesgo del sistema. Es decir, una estimación de la afectación de los activos, para descubrir aquellos que se encuentren más vulnerables, lo cuál ayudará al momento de decidir como priorizar los requerimientos de seguridad y así minimizar los riesgos de la aplicación.

En esta fase de estimación de riesgos, se estima el impacto al que están expuestos los activos del sistema, se tomó en cuenta el impacto potencial al que está expuesto el sistema teniendo en cuenta el valor de los activos y la valoración de las amenazas.

Para los cálculos del valor del activo se utilizó la siguiente escala:

- **B:** Bajo
- **M:** Medio
- **A:** Alto

Por otra parte, para valoración de las amenazas a las que esta expuesto el sistema se realizó una priorización de los casos de abuso, se decidió clasificarlos de acuerdo al efecto que tendría sobre todo el sistema. Se realizó un concentrado para evaluar cada caso de abuso. Los niveles de valoración de las amenazas se basaron en una escala de 10 de acuerdo a los siguientes rangos: 1-3= Baja, 4-6= Media, 7-10 = Alta. La tabla 4.6 muestra la clasificación de los casos de abuso:

Caso de Abuso	Rango	Valoración de las amenazas
MC-01	10	Alta
MC-02	7	Alta
MC-03	7	Alta
MC-04	3	Baja
MC-05	3	Baja
MC-06	5	Media
MC-07	3	Baja
MC-08	5	Media
MC-09	3	Baja
MC-10	5	Media
MC-11	8	Alta
MC-12	5	Media

Cuadro 4.6: Priorización de los casos de abuso

Además, para el cálculo del riesgo, se utilizó la notación de la Tabla 4.2

Riesgo		**Posibilidad de Ocurrencia**		
Valor del Activo		**Alta**	**Media**	**Baja**
	A	A	A	M
	M	A	M	M
	B	M	B	B

Figura 4.2: Cálculo de impacto

Tomando en cuenta el valor de los áctivos y la valoración de las amenazas de los casos de los casos de abuso, se realizó el cálculo de estimación de impacto, el resultado se encuentra concentrado en la Tabla 4.7. Para el cálculo de la posibilidad de ocurrencia se calculo el promedio de acuerdo a los valores númericos de priorización de los casos de abuso presentados en la tabla 4.6.

Activo	**Propiedad**	**Valor**	**Caso de Abuso**	**Posibilidad de ocurrencia**	**Impacto**
CFD	Disponibilidad	A	MC-02	7 Alta	A
CFD	Integridad	A	MC-06,MC-11	6.5 Media	A
CFD	Confidencialidad	M	MC-02,MC-11	7.5 Alta	A
Llave privada	Disponibilidad	A	MC-01,MC-02	8.5 Alta	A
Llave privada	Integridad	A	MC-03	7 Alta	A
Llave privada	Confidencialidad	A	MC-01,MC-02	8.5 Alta	A
Folios	Disponibilidad	M	MC-01,MC-02,MC-03,MC-07	6.75 Alta	A
Folios	Integridad	M	MC-03	7 Alta	A
Folios	Confidencialidad	M	MC-01,MC-02	8.5 Alta	A
Certificado digital	Disponibilidad	A	MC-01,MC-02,MC-03,MC-07	6.75 Alta	A
Certificado digital	Integridad	A	MC-03,MC-05	5 Media	A
Certificado digital	Confidencialidad	B	MC-01,MC-02	8.5 Alta	M
Cartera de clientes	Disponibilidad	M	MC-06,MC-07	4 Media	M
Cartera de clientes	Integridad	M	MC-06,MC-07,MC-09,MC-11	4.75 Media	M
Cartera de clientes	Confidencialidad	A	MC-06,MC-07,MC-09	3.66 Media	A
Reporte mensual	Disponibilidad	M	MC-06	5 Media	M
Reporte mensual	Integridad	M	MC-06,MC-10	5 Media	M
Reporte mensual	Confidencialidad	M	MC-06,MC-10	5 Media	M
Aplicación	Disponibilidad	A	MC-03,MC-04,MC-06,MC-07,MC-09	4.2 Media	A
Aplicación	Integridad	A	MC-02,MC-03	7 Alta	A
Aplicación	Confidencialidad	B			B
Dispositivo	Disponibilidad	A	MC-06	5 Media	A
Dispositivo	Integridad	A	MC-01,MC-02,MC-03,MC-04,MC-05,MC-08,MC-09,MC-11	5.75 Media	A
Dispositivo	Confidencialidad	B		Baja	B
Servidor	Disponibilidad	A	MC-06	5 Media	A
Servidor	Integridad	A	MC-12	5 Media	A
Servidor	Confidencialidad	M	MC-12	5 Media	M

Cuadro 4.7: Estimación de riesgo

4.2.4. Resultados de la metodología SQUARE Lite

Como resultado de la aplicación de las etapas desarrolladas durante el trabajo realizado con el uso de la metodología SQUARE Lite, se han podido identificar y dar prioridad a los requerimientos de seguridad. Se desarrolló una clasificación dividido de acuerdo a las siguientes métricas:

- **Esenciales:** implica que el producto no será aceptable a menos que estos requerimientos se cumplan.
- **Condicionales:** implica que estos requerimientos podrían mejorar el producto, pero el producto no sería inaceptable si estos están ausentes.
- **Opcionales:** implica una clase de funciones que pueden o no ser completamente necesarias.

Cada requerimiento fue clasificado de acuerdo a las métricas de arriba. Los requerimientos esenciales todos corresponden directamente a las amenazas más altas realizadas al resultado de estimación de riesgo.

Requerimientos Esenciales de Seguridad

En nuestro análisis, estos 12 requerimientos de seguridad fueron considerados como esenciales:

RS-1 Definición de Arquitectura de Información: Describir el proceso de arquitectura de información para ubicar la lógica de negocio y las métricas de seguridad.

RS-2 Control de Acceso: Para ingresar al sistema es necesario introducir el nombre de usuario y password. El password que se introduzca debe de estar basado en un estándar deberá de contener letras, números y caractéres especiales. Datos de la aplicación como la contraseña de la aplicación debe de ser secreto.

RS-3 Blanqueado de memoria: Parámetros críticos de seguridad como la llave privada y la contraseña de acceso deben de ser blanqueados de memoria cuando no sean utilizados.

RS-4 Protección de parámetros críticos de seguridad: Parámetros críticos de seguridad como la llave privada debe de ser protegida, de acceso no autorizado, divulgación, modificación o sustitución.

RS-5 Protección de parámetros públicos de seguridad: Parámetros públicos de seguridad como los certificados digitales y folios deben de ser protegidos, de acceso no autorizado, divulgación, modificación o sustitución.

RS-6 Autoprueba del sistema: La aplicación debe de realizar una función de autoprueba para comprobar la integridad de parámetros como: llave privada, folios y cartera de clientes.

RS-7 Canal seguro: Parámetros públicos de seguridad tales como la cartera de cliente y los comprobantes fiscales digitales deben de ser transportados electrónicamente através de un canal seguro.

RS-8 Validación de entradas y sálidas: Se debe de hacer una validación de todas las entradas y salidas de la aplicación una vez que la aplicación se encuentre en un ambiente operativo.

RS-9 Protección de ataques físicos: Mediante un respaldo de la información sensitiva en el dispositivo se tendrá que respaldar la información en un servidor.

RS-10 Almacenamiento Seguro: Se debe de buscar que las facturas electrónicas se guarden en un almacenamiento seguro en el servidor. Despues de que la información sea respaldada, la cartera de clientes debe de ser cifrada en el servidor seguro.

RS-11 Protección de datos ante aplicaciones externas: Otras aplicaciones no podrán acceder a los datos de la aplicación de facturación electrónica. Esto es, otras aplicaciones no podrá acceder a los datos como CFD's, certificados digitales, folios digitales, información de los clientes y contraseña que se encontraran guardados en la memoria SD.

RS-12 Bloqueo de la aplicación: Después de cierto tiempo la aplicación debe de bloquearse para evitar que usuarios no autorizados no acceda a información privada de usuario o generen facturas sin consentimiento del contribuyente

Estos 12 requerimientos abordan directamente las dimensiones de seguridad de integridad, autenticación, confidencialidad, disponibilidad y amenazas físicas. Consideramos que estos requerimientos de seguridad son esenciales para el sistema de facturación electrónica, sin cumplir estos requerimientos, no se podría considerar que el sistema es seguro ante las principales amenazas que afecten al sistema. Estos requerimientos deben de estar impactados en el diseño e implementación del sistema

Requerimientos Condicionales de Seguridad

En nuestro análisis, estos 5 requerimientos de seguridad fueron considerados como condicionales. Cumplir con estos requerimientos de seguridad añadiran cierta seguridad al sistema de facturación electrónica pero no son necesariamente esenciales para defender la aplicación de facturación electrónica:

RS-13 Firma de la aplicación: El código ejecutable de la aplicación debe de ser firmado.

RS-14 Servidor: El servidor que se utilice para realizar el respaldo de CFD's debe ser protegido ante ataques pasivos o activos.

RS-15 Uso de técnicas de diseño e implementación seguras: Durante el desarrollo del proyecto se hará uso de técnicas de modelado de seguridad.

RS-16 Uso de Buenas prácticas de codificación: Para la implementación de la aplicación se debe de usar buenas prácticas de codificación segura en el lenguaje JAVA.

RS-17 Evitar ataques DoS: El servidor siempre debe se estar disponible cuando la aplicación lo requiere (Evitar ataques de tipo DOS).

Debido a que en este trabajo el caso de estudio es una aplicación móvil de facturación electrónica, la protección del servidor es considerada como condicional sin embargo es conveniente la implementación de estos requerimientos.

El requerimiento *RS-15* y *RS-16* debería ser considerado ya que el uso de mejores prácticas pueden disminuir las vulnerabilidades a las que está expuesto el sistema.

Requerimientos Opcionales de Seguridad

En nuestro análisis, estos 3 requerimientos de seguridad fueron considerados como opcionales:

RS-18 Actualización del Servidor: El servidor debe ser actualizado con cierta frecuencia (2 Meses).

RS-19 Políticas de servidor: Se deben de bloquear todos los puertos, protocolos y servicios que no sean necesarios en el servidor.

RS-20 Roles de Usuario: En el sistema deben de existir dos perfiles de usuario:

- Perfil de usuario de confianza: Usuario común que podrá realizar las funciones de facturación electrónica, envio de reportes mensuales, configuraciones básicas.
- Perfil de administrador: Usuario que podrá realizar funciones avanzadas.

Estos requerimientos de seguridad agregarían cierta seguridad al sistema de facturación electrónica. En base a nuestra estimación de riesgos se determinó que estos requerimientos son opcionales, y no son necesarios. Por ejemplo el requerimiento de seguridad RS-20, no es necesario ya que el uso de los dispositivos móviles es individual, la segregación de funciones resulta complicada, sin embargo considerar este requerimiento podría ser considerado como condicional en un sistema de facturación electrónica de mayor escala.

4.3. Arquitectura

4.3.1. Criterios de diseño

Debido a que desarrollo de este trabajo se hará bajo la plataforma Android y dada la naturaleza de la aplicación de facturación electrónica, donde la lógica de negocio puede variar con el tiempo y las reglas de negocio podrán modificar a las actuales y nutrirse de otras nuevas. Es por eso que el modelado se hará utilizando programación orientada a objetos, formando así objetos muy similares a la realidad que se rigen por las reglas de negocio.

Para poder acompañar los cambios del negocio, actualizando así el modelo del dominio, se buscó la manera de mantener este dominio lo mas aislado posible del resto de la aplicación. Para ello la arquitectura elegida es una arquitectura basada en capas lógicas (*Layer Pattern*). Buscando que esta arquitectura sea escalable y lo más segura posible, es deseable poder distribuir la arquitectura en capas lógicas y poder buscar el mayor desacoplamiento entre ellas. En la Figura 4.3 podemos ver como quedaron distribuidas las diferentes partes de la aplicación sobre la arquitectura propuesta.

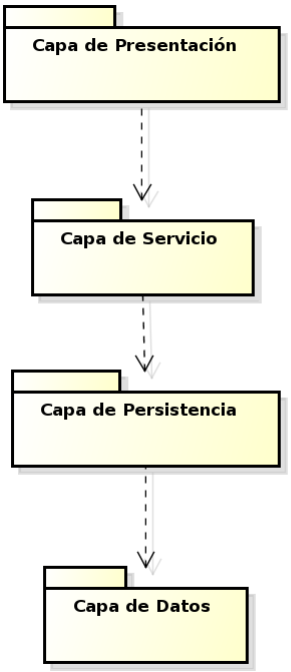


Figura 4.3: Arquitectura en capas

4.3.2. Análisis del diseño

Definida la arquitectura a utilizar, a continuación se analiza el diseño de cada capa del diseño.

Capa de presentación

Esta es la capa que interactua con el usuario final, es la encargada de presentar la información y recolectarla para hacer uso de los servicios expuestos por la capa de servicio, para satisfacer los casos de uso de la aplicación.

En este trabajo como se mencionó con anterioridad se hará bajo la plataforma Android. Android utiliza la clase **Activity** para delegar la capacidad de presentación. Las actividades de Android son esenciales ya que forman las pantallas de la aplicación y desempeñan un papel en el ciclo de vida de una aplicación en Android. Las actividades están formadas por componentes secundarios denominados vistas. Las vistas son lo que ven los usuarios y con lo interactúan. Controlan el diseño, proporcionan elemento de texto para etiquetas e información, botones y formularios para entradas del usuario, y dibujan gráficos en la pantalla. También se utilizan para registrar oyentes de la interfaz, como los de controles de pantallas táctiles. Las vistas y otros componentes de Android utilizan cadenas, colores, estilos y gráficos a los cuales se conocen como recursos. La figura 4.4 muestra la relación entre actividades vistas y recursos.

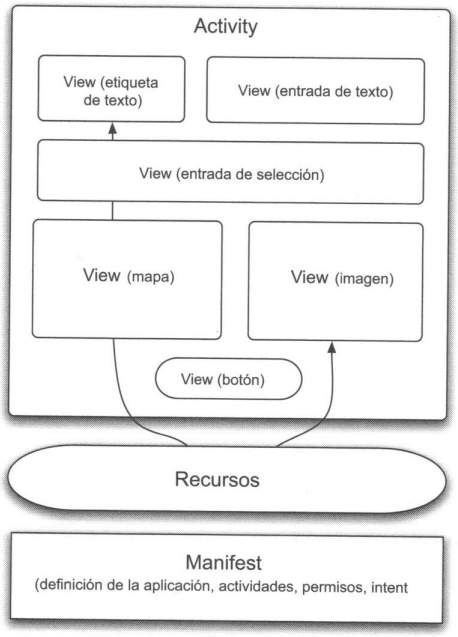


Figura 4.4: Diagrama de relación entre Activity, View, recursos y manifesto

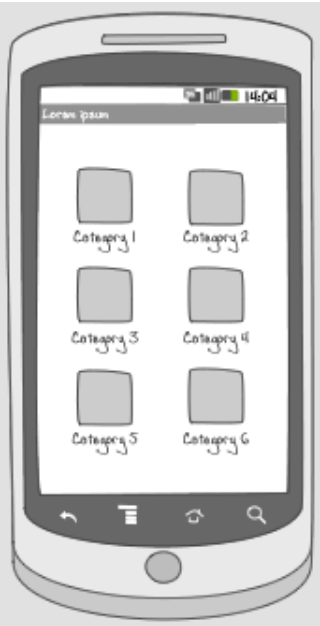


Figura 4.5: Ejemplo de dashboard

Por otra parte para optimizar el diseño de la interfaz de usuario se usaron los siguientes patrones de diseño de interfaces de usuario:

- **Dashboard:** Un dashboard es una pantalla de bienvenida de la aplicación dando un punto de entrada para la el usuario. El dashboard puede ser estático o dinámico. En la pantalla del dashboard se muestra las acciones importantes o que se usaran con frecuencia en la aplicación en forma de categorías. Las categorías son representadas con un icono y su título y se encuentran en pantalla completa en una orientación de grid. Este patrón permite que el usuario encuentre el contenido de una forma más fácil. Un ejemplo de dashboard se puede ver en la figura 4.5

- **Barra de acción** La barra de acción se encuentra localizada en la parte alta de la pantalla y muestra funcionalidades importantes, por lo regular se ocupa para realizar opciones de búsqueda, refrescar datos y composición. Un ejemplo del patrón barra de acción puede verse en la figura 4.6.

- **Barra de búsqueda:** La barra de búsqueda esta anclada en la parte de arriba de la pantalla y remplaza la barra de acciones. Contiene 3 elementos: un campo de texto permitiendo al usuario introducir los parámetros de búsqueda, el botón de búsqueda y un selector de popup para múltiples búsquedas. Este patrón será conveniente adoptarlo para nuestro sistema de facturación electrónica ya que se podrán realizar búsquedas múltiples. Un ejemplo del patrón barra de busqueda puede verse en la figura 4.7.

Figura 4.6: Ejemplo de barra de accion

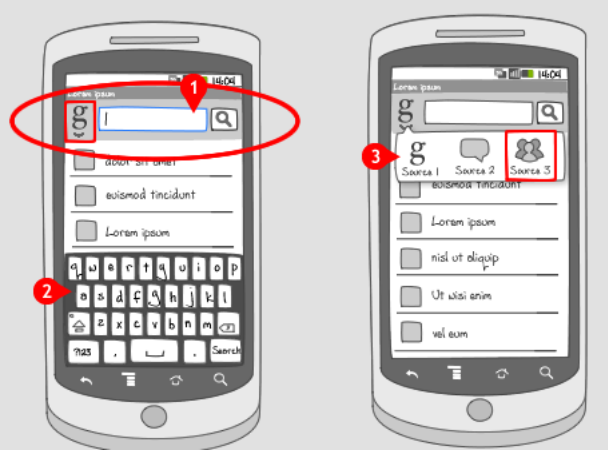


Figura 4.7: Ejemplo de barra de busqueda

Capa de servicios

La capa de servicios es la que se encarga de brindar los servicios necesarios a la capa de presentación. En nuestro caso esta capa se encargará de facilitar la creación de facturas electrónicas. Para el diseño de esta capa se hizo uso del patrón de diseño fachada. El patrón fachada provee una interfaz unificada a un conjunto de interfaces en un subsistema, definiendo una interfaz de más alto nivel para que el subsistema sea más fácil de usar.

Los objetos intercambiadas entre las capas de servicio y de presentación serán objectos del tipo Data Transfer Object, que son simples objetos simples (POJO) utilizados en el intercambio de información.

En el diagrama de clases de la figura 4.8 se puede ver el diseño para la creación de facturas electrónicas usando el patrón de diseño fachada.

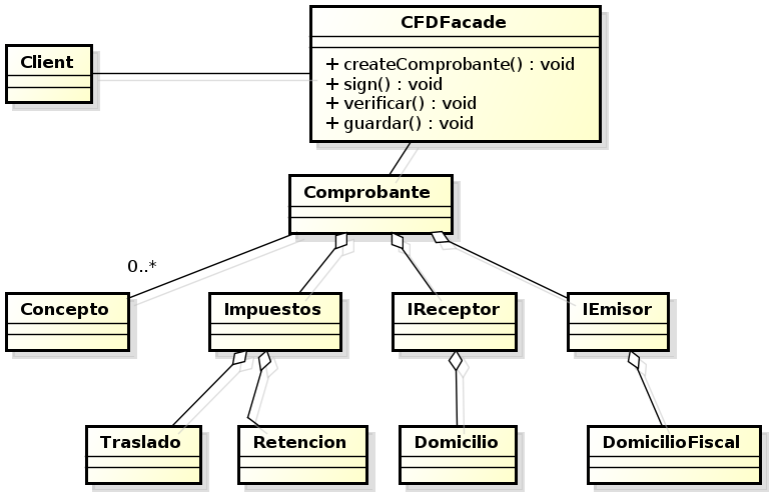


Figura 4.8: Patrón fachada para cfd

Capa de persistencia

La capa de persistencia es la encargada de abstraer y resolver el acceso a datos en cualquier modelo de base de datos. El objetivo es ser la única que conoce como son persistidos los objetos de la aplicación y como recuperarlos abstrayendo las impedancias entre objetos y la fuente de datos.

La capa superior o de vista no interactúa directamente con la base de datos, si no que lo hace mediante la interfaz expuesta por la capa de persistencia. Esto permite cambiar la estrategia con que se persisten los objetos, incluso cambiar la tecnología o el motor utilizado sin impactar más allá de ésta capa.

Algunos de los requerimientos buscados por la capa de persistencia son los siguientes:

- Encapsular los mecanismos de persistencia (utilizando métodos del estilo: save(obj), delete(obj),create(obj), find(obj)).
- Identificación de objetos.
- Manejo de transacciones.
- Posibilidad de realizar consultas.

En la Figura 4.9 puede observarse el mecanismo de persistencia, donde una clase de la capa de presentación solicita una búsqueda con ciertos parámetros a partir de un criterio dado.

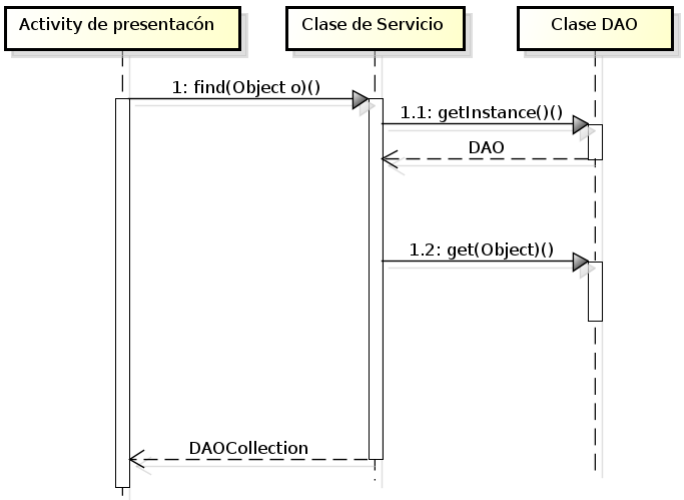


Figura 4.9: Diagrama de secuencia: persistencia de datos

Por otra parte se hace uso del patrón de de diseño Data Access Object(DAO) que nos permitirá una interfaz abstracta através de un mecanismo de persistencia sin exponer los detalles de la base de datos. En la figura 4.10 se muestra la vista lógica de como se aplicó este patrón de diseño através de este diagramá de clases. Como mecanismo para guardar datos se utilizará una base de datos XML.

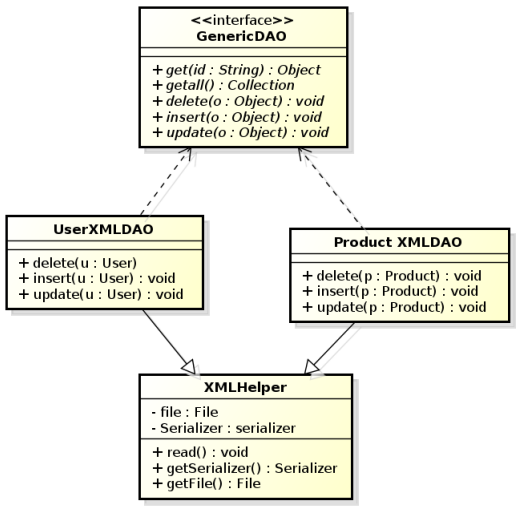


Figura 4.10: Patrón de persistencia DAO

La clase XMLHelper presentada en la figura 4.10 nos ayudará a leer estos datos del documento XML y deserializarlos en objetos POJO que contienen la lógica de negocio del sistema de facturación electrónica y así poderlos manipularlos de manera más sencilla. Para la serialización de documentos XML a objetos POJO se hizó uso de la bibloteca SIMPLE [6].

4.3.3. Arquitectura de seguridad

Una arquitectura de seguridad describe como el sistema conjunta mecanismos de seguridad apropiados para satisfacer los requerimientos de seguridad, y de esta forma proteger el sistema ante amenazas. En este trabajo se ha realizado una arquitectura de seguridad donde se definió un conjunto de componentes de seguridad que cubren la mayoría de los requerimientos de seguridad especificados con la metodología SQUARE. Esta arquitectura de seguridad protege al sistema de facturación electrónica y ofrece el soporte necesario para asegurar que los requerimientos de seguridad sean cubiertos.

Esta arquitectura de seguridad debe de ser integrada a la arquitectura propuesta anteriormente añadiendo una nueva capa de seguridad que contribuye con aspectos de seguridad convirtiendo esta arquitectura de software convencional en una arquitectura de software segura que será implementada subsecuentemente. La arquitectura final del sistema de facturación electrónica agregando una capa de seguridad adicional quedaría de la siguiente manera como se muestra en la figura 4.11.

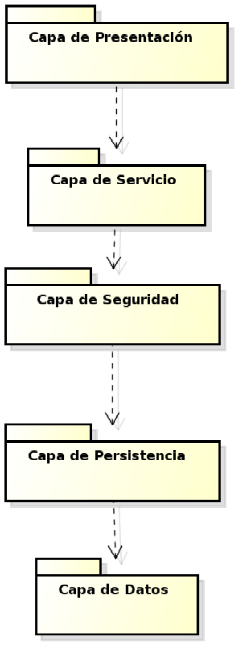


Figura 4.11: Componentes de seguridad

Los componentes de seguridad de esta capa de seguridad serán definidos a través de la extensión UMLSEC. Estos componentes de seguridad servirán como referencia para cualquier diseño aplicaciones móviles de facturación electrónica e incluso un subconjunto de estos componentes podrán servir para aplicaciones basadas en escritorio o aplicaciones web de facturación electrónica. A partir de los requerimientos de seguridad obtenidos por la metología SQUARE se realizó el modelado de componentes utilzando la herramienta de seguridad UMLSEC. A continuación se muestra como se realizó el diseño de los componentes de seguridad para el sistema de facturación electrónica.

Canal Seguro

Para el modelado de este componente se consideran los requerimientos referentes a proteger las comunicaciones entre el dispositivo móvil y el servidor de respaldo. El diseño corresponde a los siguientes requerimientos *RS-7*, *RS-9* y *RS-10* de nuestra fase análisis de requerimientos.

Podemos ver que en estos requerimientos necesitamos proporcionar privacidad entre los elementos, es decir se requiere preservar la privacidad entre los datos enviados entre el dispositivo móvil y el servidor de resplado y una vez que la información como la cartera de clientes llegue al servidor está información debe ser protegida. Bajo estas directivas se aplicó el estereotipo << *data security* >> que es usado para asegurar que los requerimientos de seguridad dados por los estereotipos <<

critical >> y las etiquetas asociadas se cumplan. La figura 4.12 da una especificación de un subsistema de comunicaciones de un objeto emisor (dispositivo móvil) a un objeto receptor (servidor de respaldo). En el subsistema, el objeto emisor supone aceptar un valor d que corresponde a los datos que serán enviados (comprobante fiscales, cartera de clientes) al objeto receptor. Los datos enviados del dispositivo móvil será enviado de forma cifrada (<< *encrypted* >>) por medio de la red. De acuerdo al estereotipo << *critical* >> y la etiqueta {*secrecy*} se guardará la privacidad de los datos enviados. Para este componente se definió un adversario por defecto, las acciones que puede realizar el adversario estan resumidas en la tabla 4.8.

Stereotype	**Threats**$_{default()}$
Internet	delete,read, insert
encrypted	delete

Cuadro 4.8: Acciones del adversario

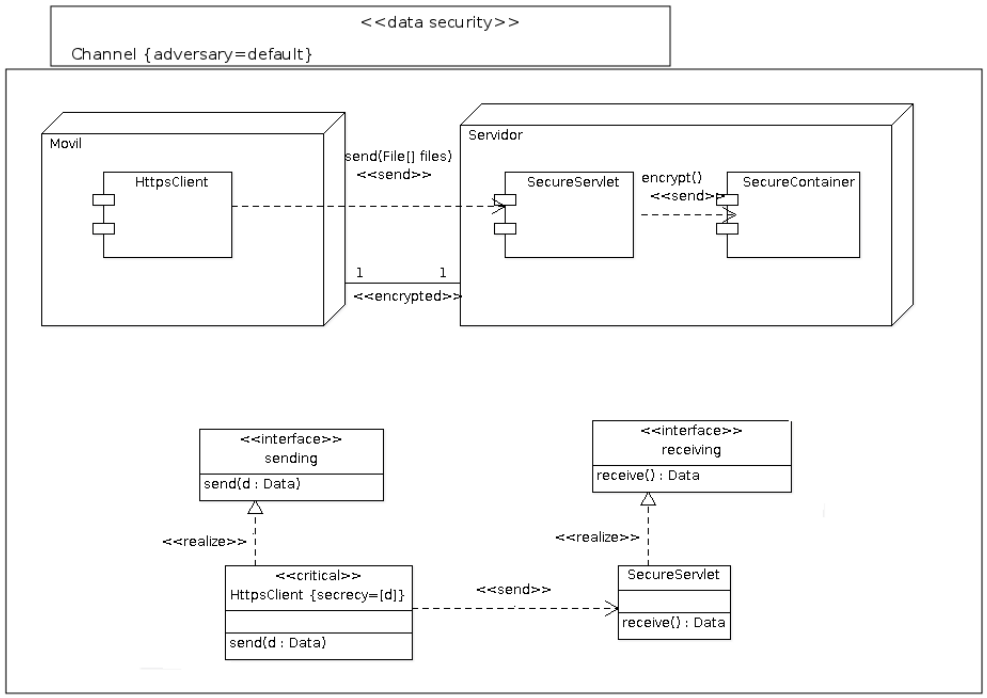


Figura 4.12: Subsistema de comunicaciones

Control de acceso

El control de acceso se refiere a proteger el acceso a la aplicación ante personas no autorizadas, para realizar el acceso al sistema este se hará através de un sistema de autenticación basado en algo conocido, en este caso será el nombre de usuario y una contraseña. La interacción para lograr la autenticación se muestra en la figura 4.13 en el diagrama de secuencia.

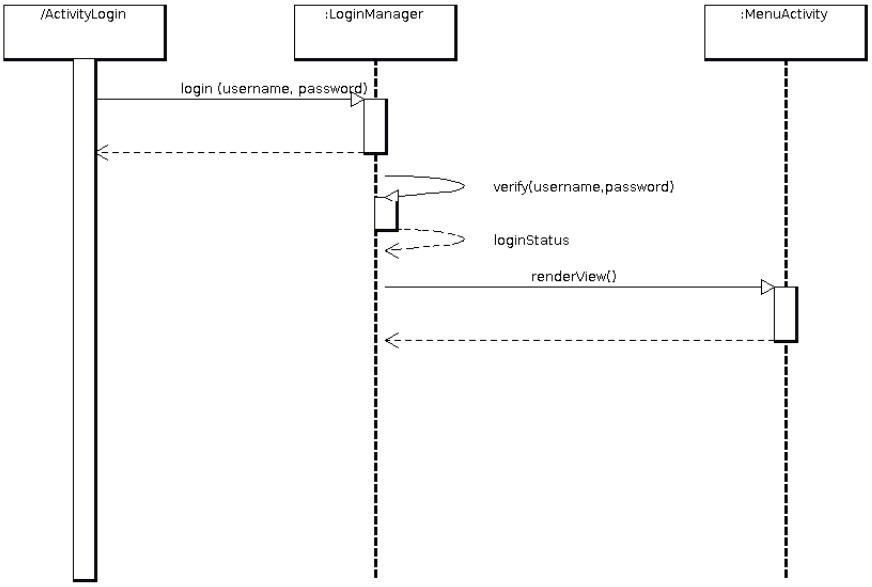


Figura 4.13: Diagrama de Secuencia: control de acceso

El diseño de este componente cubre el requerimiento de seguridad *RS-2* donde el elemento de autenticación como el password de usuario debe mantenerse secreto. Para cubrir este requerimiento se utilizará el estereotipo *guarded access* el cuál nos indica que cada objecto en el subsistema estereotipado como $<< guarded >>$ solo puede ser accedido por medio de objetos que tengan la etiqueta $\{guard\}$ ligados al objeto $<< guarded >>$. En nuestro caso el objecto el cuál se debe de mantenerse guardado es el password por lo tanto deberá ser etiquetado de la siguiente forma $\{guard = password\}$. En la figura 4.14 se puede ver el modelado de este componente por medio del diagrama de clases.

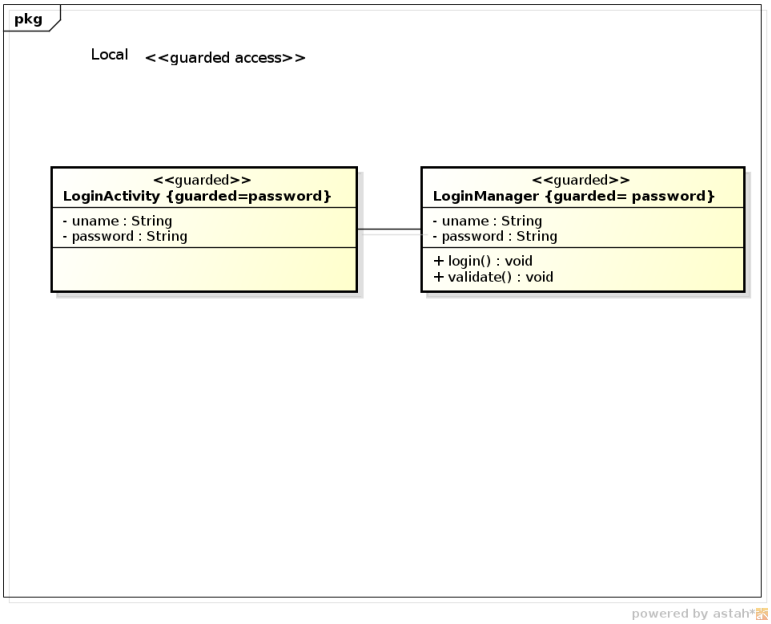


Figura 4.14: Diagrama de Clases: control de acceso

Contenedor de seguro

El componente de contenedor seguro será el encargado de mantener los datos sensitivos de la aplicación confidenciales, para que personas no autorizadas no puedan acceder a este contenido. Un ejemplo de uso del contenedor seguro se muestra en la figura 4.15, donde una vez que se realiza alguna acción (inserción, modificación, borrado) sobre la capa de persistencia de nuestra aplicación, los datos deben de ser guardados y cifrados en un contenedor seguro.

El diseño de este componente cubre los requerimientos *RS-4*,*RS-5*, *RS-10* y RS-11 de nuestra fase análisis de requerimientos. Para el diseño de este subsistema se hará uso del estereotipo $<< no\ down - flow >>$ el cuál pretende prevenir de fuga de información o corrupción de los datos. Este esteoreotipo permite dar especial atención en el flujo de información segura con el uso de la etiqueta $\{high\}$ asociado con el estereotipo $<< critical >>$ que nos ayudará a proteger la fuga de los datos marcados con esta etiqueta. El diseño de este componente se puede ver en el diagrama de clases de la figura 4.16.

Autoprueba del sistema

El componente de autoprueba del sistema se encargará de verificar que datos sensitivos, como lo es la llave privada, certificados, cartera de clientes o folios no sean modificados, en caso de que alguno de estos componentes se haya alterado, el

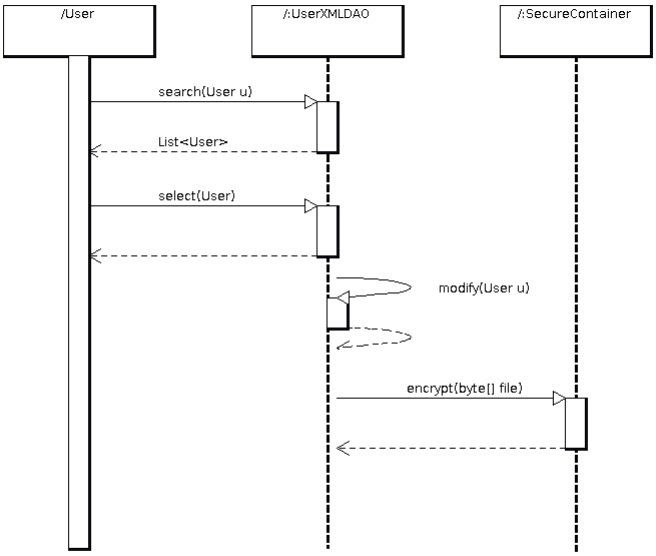


Figura 4.15: Diagrama de secuencia: contenedor seguro

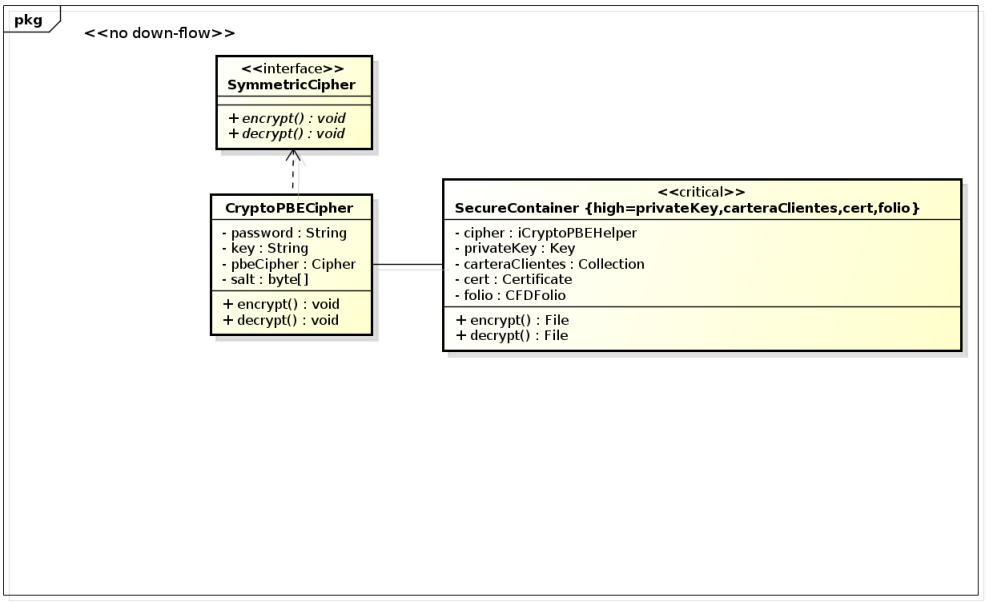


Figura 4.16: Diagrama de clase: contenedor seguro

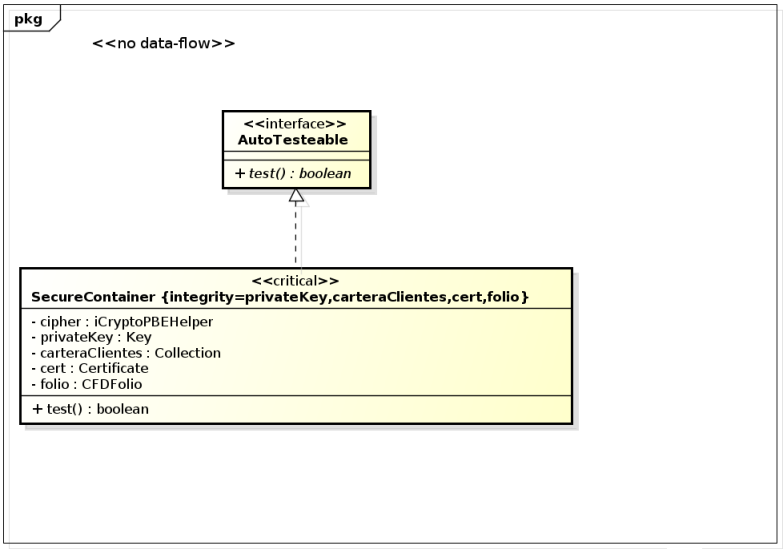


Figura 4.17: Diagrama de clase: Autoprueba

sistema denegará el acceso a la aplicación. El diseño de este componente impacta directamente al requerimiento de seguridad *RS-6*. En el diseño de este sistema se hizó uso del estereotipo $<< no\ down - flow >>$ en este caso lo que se intenta con este estereotipo es que no exista corrupción en los datos manteniendo así la integridad de los datos con la etiqueta $\{integrity\}$ asociado con el estereotipo $<< critical >>$. El diseño de este componente se puede ver en el diagrama de clases de la figura 4.17.

Firma electrónica

El componente de firma de electrónica será el encargado de firmar la cadena original del comprobante fiscal digital. El diseño de este componente no corresponde a ningún requerimiento de seguridad de nuestra fase de análisis de requerimientos de seguridad sin embargo es un requerimiento funcional de la aplicación. Para el diseño de este componente se usará el estereotipo $<< critical >>$ que se puede usar con la etiqueta $\{authenticity\}$ que nos servirá para mostrar la autenticidad del origen de la factura electrónica. En la figura 4.18 se puede ver como fue el diseño de este componente.

Bloqueo de la aplicación

Este componente de seguridad debido a su naturaleza no requiere ser modelado con la extensión UMLSEC, ya que no contiene manejo de información sensitiva sin embargo se considera un componente de seguridad ya que nos permite que no exista

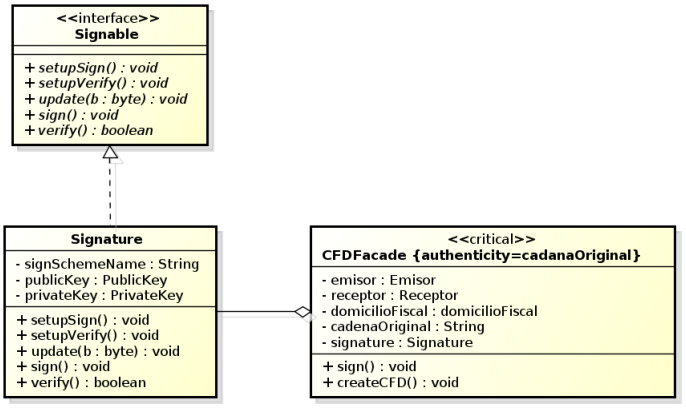


Figura 4.18: Diagrama de clase: Firma electrónica

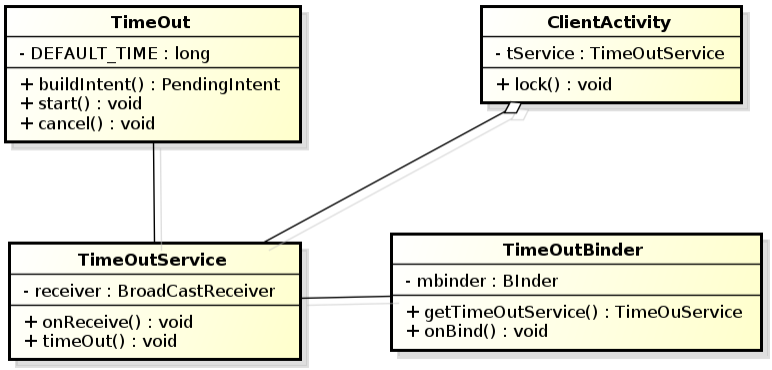


Figura 4.19: Diagrama de clase: Bloquear aplicación

fuga de información cuando el dispositivo no se encuentre en uso. En la figura 4.19 el diseño de este componente.

Capítulo 5

Implementación y pruebas

En capítulos anteriores se ha mostrado el modelo operativo de un sistema de facturación electrónica, explicando tanto los requerimientos legales del Sistema de Administración Tributaria, requerimientos funcionales, requerimientos de seguridad y una arquitectura de seguridad para este tipo de aplicaciones. En base a estos parámetros en este este capítulo se hará una descripción de la implementación de los componentes de seguridad de para así formar así una aplicación segura de facturación electrónica.

5.1. Codificación Segura

Desarrollar aplicaciones robustas de software que sean libre de vulnerabilidades es una tarea díficil, y hacerlas totalmente seguras es imposible. Sin embargo el uso de prácticas de codificación segura que nos puede ayudar significativamente a reducir defectos, vulnerabilidades e errores de software. Las prácticas de codificación segura requiere un entendimiento de los errores de programación que pueden llevar a vulnerabilidades. Dentro de este trabajo se siguieron las siguientes prácticas de codificación segura:

- Asegurar que las variables se inicialicen correctamente, no depender de inicialización por defecto [12].
- Comprobar explícitamente todas las llamadas al sistema o los valores de retorno de métodos o funciones [12].
- Asegurar que los archivos no se puedan abrir utilizando rutas relativas de archivos o referencias indirectas de archivo [12].
- Evitar la invocación de programas de menos de confianza, como shells de comandos, desde el interior de nuestra aplicación [12].

- Evitar el uso de generadores de números pseudo-aleatorios en su lugar, utilizar una API para generar números pseudo-aleatorios de una manera criptográficamente segura [12, 13].

- Poner en práctica el manejo de excepciones de forma segura y asegurar que la información sensible no se filtre [13].

- Proteger datos secretos mientras se encuentran almacenados en la memoria (llaves criptográficas, contraseñas) [13].

- Evitar el *hard-coding* de datos secretos (por ejemplo, contraseñaas, llaves criptográficas), cifrar y almacenarlos en archivos externos [13].

- Usar enunciados parametrizados en las consultas de base de datos (para evitar inyección SQL) [13].

- Eliminar el código que es obsoleto o que no se utiliza [12, **?**].

Adicionalmente a estas prácticas se han adoptado algunas otras que son parte del propias del lenguaje Java. Estas prácticas de codificación segura en Java pueden encontrarse en [3].

5.1.1. Revisión de código

La revisión de código ocupa un lugar importante en la lista de buenas prácticas que están destinadas a mejorar la seguridad del software. Para ayudarnos a realizar esta tarea existen herramientas para verificar errores e inconsistencias en el código antes de compilarse, a este tipo de herramientas se les conoce como herramientas de análisis estático.

Para realizar la revisión de código se utilizó la herramienta **FindBugs** que es una herramienta de código estático que nos permite visualizar errores y violaciones en código Java. La herramienta **FindBugs** se ejecutó al concluir la codificación del proyecto, mostrando alrededor de 174 errores como se muestra en la figura 5.1. Dentro de los cuales 76 son de prioridad alta, 68 de prioridad media y 30 de prioridad baja.

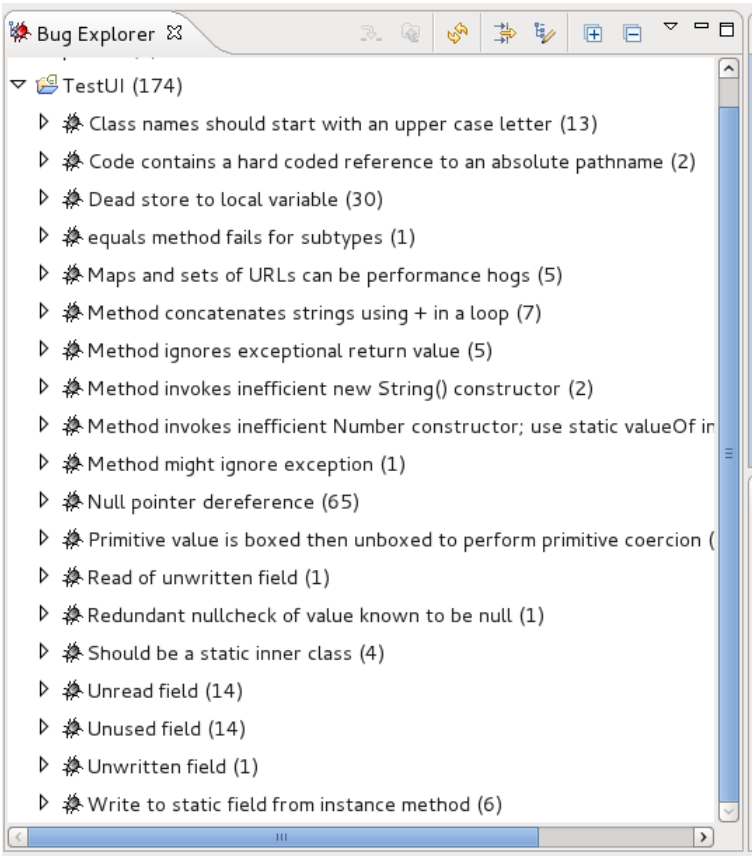


Figura 5.1: Errores encontrado por FindBugs

Al analizar el código con esta herramienta, se eliminaron la mayoría de los errores que fueron encontrado por la herramienta tal como lo muestra la figura 5.2. Quedando un total de 16 de errores.

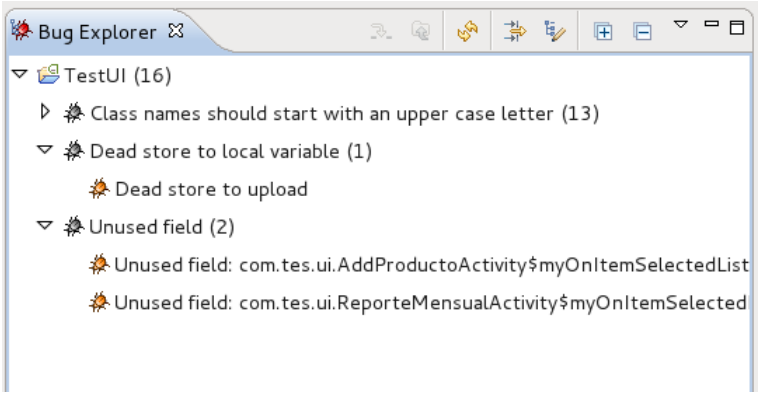
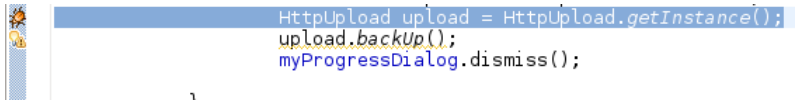


Figura 5.2: Revisión de código FindBugs

Estos 16 errores se consideraron como falsos positivos. A continuación se dará un listado explicando porque estos errores son falsos positivos:

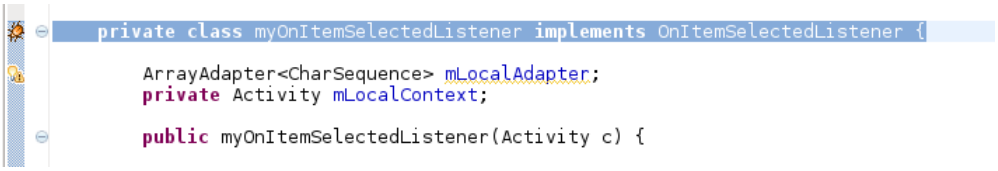
- **Los nombre de las clases deben comenzar con una letra mayúscula** A pesar de que esto se considera un estándar de codificación en la plataforma Java. Android autogenera clases para acceder sus recursos declarando clases del estílo `public start final class anim`.

- **Variable local muerta**: Este error se refiere a que una variable local es asignada, pero su valor no es leído ni utilizado en ninguna instruccón subsecuente. Al revisar el código manualmente se encontró que esta variable se utilizaba posteriormente en el código.



```
HttpUpload upload = HttpUpload.getInstance();
upload.backUp();
myProgressDialog.dismiss();
```

- **Campos no utilizados:** Este parámetro se refiere a eliminar parámetros que no se utilizan. En nuestro caso se refiere al no uso de dos clases internas, sin embargo estas clases internas son partes del funcionamiento de este módulo.



```
private class myOnItemSelectedListener implements OnItemSelectedListener {

    ArrayAdapter<CharSequence> mLocalAdapter;
    private Activity mLocalContext;

    public myOnItemSelectedListener(Activity c) {
```

```
OnItemSelectedListener spinnerListener = new myOnItemSelectedListener(this);

s.setOnItemSelectedListener(spinnerListener);
```

5.2. Implementación de componentes de seguridad

En la siguiente sección se explicará como se realizó la implementación de los componentes de seguridad que fueron propuestos en nuestro diseño arquitectural para mitigar en medida de lo posible los ataques que pudiera sufrir una aplicación de facturación electrónica.

5.2.1. Control de acceso

Para la implementación del control de acceso se realizó através de ingresar un nombre de usuario y una contraseña. Para la creación de esta contraseña se ha decidido adoptar un conjunto de reglas que ayudaran a los usuarios a generar contraseñas seguras. Para que una contraseña se considere segura se debe de considerar lo siguiente [1]:

- La contraseña debe de contener al menos tres de los siguientes tipos de caracteres:
  - Letras minúsculas
  - Letras mayúsculas
  - Números
  - Caracteres especiales (`|@#$%^&*()_+~-=\‘{}[]:";’<>|`)
- La contraseña debe de contener al menos quince caracteres alfanúmericos.

Para nuestro caso se ha implantado que nuestra contraseña contenga letras minúsculas, al menos una letra mayúscula, al menos un número y al menos un carácter especial. La longitud de la contraseña será de al menos 8 caracteres ya que debido a la naturaleza de los dispositivos móviles resulta cansado en terminos de usabilidad el ingresar una contraseña de mayor longitud.

Por otra parte, el almacenamiento del nombre de usuario y contraseña se hará en una pequeña base de datos creada en SQLite. Para el almacenamiento de la contraseña se utilizará una función picadillo. Si no se aplica esta función picadillo a las contraseñas que se almacenan en la aplicación en caso de que un adversario comprometa la base de datos y robe la contraseña es posible comprometer a otra aplicación o servicio del usuario, si este utiliza la misma contraseña.

Al aplicar una función picadillo a las contraseña antes de que sea almacenada en la base de datos, dificultamos al adversario el determinar la contraseña original. Para el almacenamiento del campo de contraseña en la base de datos se hizo uso de el algoritmo `SHA1`.

5.2.2. Canal Confiable

El canal seguro nos servirá para enviar los datos de respaldo como nuestra cartera de clientes y los comprobantes fiscales digitales de una manera segura. Para la implementación de este canal confiable se ha decidido utilizar el protocolo **SSL/TLS**. El objetivo primordial del protocolo **SSL/TLS** es proporcionar privacidad y integridad de datos entre dos aplicaciones que se comunican. Algunos puntos clave de este protocolo son:

- **SSL/TLS** proporciona servicios de seguridad entre el protocolo **TCP** y las aplicaciones que usan **TCP**.
- **SSL/TLS** proporciona confidencialidad usando criptografía simétrica e integridad de los mensajes usando códigos de autenticación de mensajes (MAC).

- **SSL/TLS** incluye mecanismos para permitir a dos usuarios del protocolo **TCP** determinar los mecanismos de seguridad.

El protocolo **SSL/TLS** trabaja a nivel capa de transporte justo arriba del protocolo **TCP** tal como se muestra en la figura 5.3

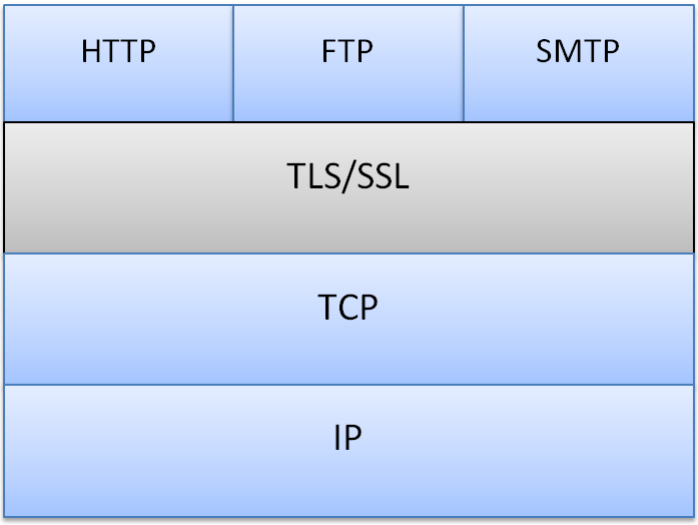


Figura 5.3: SSL/TLS

A nivel implementación existen dos aproximaciones: **SSL/TLS** puede estar integrado como parte de la suite de protocolos subyacente y por lo tanto, ser transparente a las aplicaciones o por otra parte SSL, se puede incrustar en paquetes específicos. Para nuestro caso se tomó la segunda aproximación ya que la plataforma Android nos proporciona paquetes para la abstracción de este protocolo para la implementación de un canal seguro através de **HTTPS**.

En la implementación de nuestro canal seguro se tomó como base el modelo cliente-servidor, donde nuestro dispositivo móvil servirá como cliente y realizará peticiones hacia un servidor web que prestará los servicios para guardar la información sensitiva que mande nuestro dispositivo. Para comunicarse por medio de un canal **HTTPS** es necesario la creación de un cliente HTTPS registrando la adición de los protocolos a utilizar en nuestra aplicación como se muestra a continuación:

```
public class HttpsClient extends DefaultHttpClient {

    @Override
    public ClientConnectionManager
        createClientConnectionManager(){
      final SchemeRegistry registry = new SchemeRegistry
          ();
```

```
            registry.register(new Scheme("https", new
                SSLSocketFactory(), 443));

            return new SingleClientConnManager(params, registry
                );
        }
}
```

Como puede verse se debe de sobreponer el método `createClientConnectionManager()` y en el establecer los protocolos que van a utilizarse en este caso será el protocolo https registrándolo en el esquema `createClientConnectionManager()`. Para registrar el servicio es necesario crear una fábrica de sockets SSL para el envio y recepción de información de manera segura, en nuestro caso esto se realizó heredando de la interfaz `SocketFactory` y la interfaz `LayeredSocketFactory` e implementando cada uno des sus métodos(`connectSocket, createSocket, isSecure`). A continuación se muestra como se construye esta fabrica de sockets:

```
public class SSLSocketFactory implements SocketFactory,
    LayeredSocketFactory {

  private SSLContext sslcontext = null;

  private static SSLContext createEasySSLContext() throws
      IOException {
    try {
        SSLContext context = SSLContext.getInstance("TLS"
            );
        context.init(null, new TrustManager[] { new
            EasyX509TrustManager(null) }, null);
        return context;
    } catch (Exception e) {
        throw new IOException(e.getMessage());
    }
  }

  //Implementacion de los metodos connectSocket,
      createSocket, isSecure
  ....
}
```

Por último es necesario crear un manejador de los certificados para que verifique que certificados son válidos. Para realizar esta tarea hay es necesario implementar los métodos que se heredan de la interfaz `X509TrustManager`.

```
public class AmaiTrustManager implements
    X509TrustManager {

    public void checkClientTrusted(X509Certificate[]
        chain, String authType)
                    throws CertificateException {
            ....
    }

    public void checkServerTrusted(X509Certificate[]
        chain, String authType)
                    throws CertificateException {
            ....

    }

    public X509Certificate[] getAcceptedIssuers() {
            ....
    }

}
```

5.2.3. Contenedor Seguro

El contenedor seguro es el que mantendrá confidencial la información privada de la aplicación de facturación electrónica. Para la implementación de este componente se ha decido utilizar el cifrador AES (Advanced Encryption Standard) de 128 bits. Existen muchas formas para la generación de llaves del cifrador AES como lo son el uso de números aleatorios. Sin embargo en ocasiones es conveniente usar llaves que el usuario genere por sus propios medios. Para la generación de llaves del cifrador AES se utilizará un método el cuál sea amigable con el usuario, donde para la creación de llaves se tomará como entrada la contraseña que introduzca el usuario, esta contraseña se procesa por medio de alguna función o conjunto de funciones y produce una llave adecuada para el cifrador simétrico. Al cifrado que genera las llaves de esta forma se le conoce cifrado basado en contraseña (PBE).

Los mecanismos de cifrado basados en contraseñas utilizan métodos para la prevención de ataques triviales, sin embargo un adversario podriá determinar fácilmente la contraseña si se escoge una contraseña inapropiada. Es por eso que para la creación de contraseñas se deben de usar políticas para la creación de contraseñas fuertes.

Los mecanismos de cifrado basados en contraseñas están basados en funciones picadillo. En esencia, este tipo de mecanismos utilizan la contraseña y un valor público

llamado *sal* para alimentar de alguna manera a una función de mezcla basada en una función picadillo segura aplicada un cierto números de veces. Una vez que la función de mezcla se completa, el resultado es una llave para el cifrador y posiblemente un vector de inicialización. En la figura 5.4 se muestra un esquema general de cifrado basado en contraseña.

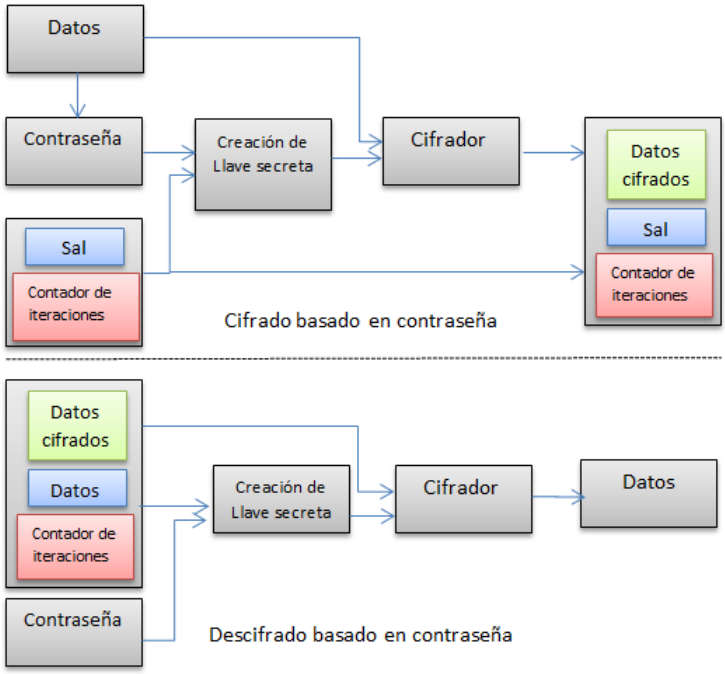


Figura 5.4: Mecanismo PBE

- **La contraseña:** Este elemento debe de mantenerse secreto y debe ser una contraseña fuerte.
- **Función de derivación de llaves (creación de llave secreta):** Una función de derivación de llaves produce una llave derivada de una llave base y otros parámetros. En una función de derivación de llave basado en contraseña, la llave base es una contraseña y los otros parámetros son el valor de la sal y el contador de iteración.
- **La sal:** Es un valor público (el adversario puede saberlo), la sal tiene como objetivo agregar una cadena de bytes aleatorios a la contraseña, una misma contraseña puede usarse para generar un gran número de llaves. Esto es útil porque obliga a los adversarios para realizar el cálculo de generación de llaves cada vez que desee probar una contraseña, por cada pieza de datos cifrados que desea atacar. La llave derivada se genera de la siguiente manera:

$$DK = KDF(C, S) \tag{5.1}$$

donde DK es la llave derivada, C es la contraseña, y S es la sal. La generación de llaves tiene 2 beneficios [19]:

1. Es difícil para un adversario precalcular todas las llaves en un diccionario de contraseñas. Si la *sal* tiene una longitud de 64 bits, habrá alrededor de 2^{64} llaves por cada contraseña. Un adversario se limita por tanto a la búsqueda de contraseñas después de que se realiza la operación basada en la contraseña dada una sal que es conocida.

2. Es poco probable que la misma llave se seleccione dos veces. Por ejemplo, si la *sal* tiene una longitud de 64 bits, la probabilidad de colisión entre llaves no se vuelve significativa hasta que cerca de 2^{32} llaves sean producidas.

- **Contador de iteración:** El contador de iteración es también un valor público. El propósito del contador de iteración es incrementar el tiempo de cálculo requerido para convertir una contraseña en una llave. Por ejemplo, si un adversario monta un ataque de diccionario, si se utiliza un contador de iteración 1000 veces en vez de una sola vez,se requerirá 1000 veces más de procesamiento para calcular la llave de la contraseña.

Para la implementación del contenedor seguro se utilizó el esquema de cifrado basado en contraseña `AES-CBC-Pad` el cuál es el algoritmo de cifrado AES con el modo de operación CBC con padding `PKCS #5` [19]. La plataforma Android nos ofrece mecanismos para generar cifrado basado en contraseña. La creación del cifrado de basado en contraseña puede dividirse en tres partes:

1. **Creación de sal:** Para la creación de la sal se ha decidio que este parámetro se haga de manera aleatoria. Para la generación de este parámetro se utilizó el generador de números pseudoaleatorio **SHA1PRNG**, que esta basado en el algoritmo de digesto `SHA-1`. Un ejemplo de generación de sal se muestra a continuación:

```
public static String generateSalt() {
    byte[] salt = new byte[8];
    SecureRandom sr;
      try {
        sr = SecureRandom.getInstance("SHA1PRNG");
        sr.nextBytes(salt);

      } catch (NoSuchAlgorithmException e) {
                    e.printStackTrace();
      }
            return toHexString(salt);
}
```

2. **Inicialización del Cifrador basado en contraseña:** En esta parte se inicializan los datos del cifrador a utilizar así como la conversión de la contraseña a una llave del cifrador. Un ejemplo de esta inicialización se muestra a continuación:

```
private void initialize(){
    pbeParamSpec = new PBEParameterSpec(salt, count);
    try {
      keyFac = SecretKeyFactory.getInstance(
         algorithmMedium,"BC");
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    } catch (NoSuchProviderException e) {
        e.printStackTrace();
    }
}
```

```
public void setPassword(String pass) {
password = pass;
pbeKeySpec = new PBEKeySpec(password.toCharArray());
try {
    pbeKey = keyFac.generateSecret(pbeKeySpec);
    pbeCipher = Cipher.getInstance(algorithmMedium,"
       BC");
    } catch (InvalidKeySpecException e) {
        e.printStackTrace();
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    } catch (NoSuchProviderException e) {
        e.printStackTrace();
    } catch (NoSuchPaddingException e) {
        e.printStackTrace();
    }
}
```

3. **Cifrar Datos:** Una vez inicializados los datos ya es posible cifrar datos. El siguiente código nos muestra como se puede realizar esta acción:

```
pbeCipher.init(Cipher.ENCRYPT_MODE, pbeKey,
   pbeParamSpec);
ciphertext = pbeCipher.doFinal(plaintext.getBytes());
```

Dentro de las clases principales que abstraen el funcionamiento del cifrador basado en contraseña se encuentran:

- `javax.crypto.spec.PBEParameterSpec`: Esta clase nos sirve como portador de sal y el contador de iteraciones para que puedan ser inicializados en nuestro cifrador.
- `javax.crypto.spec.PBEKeySpec`: Esta clase es capaz de llevar a la sal, el contador de iteraciones y, si es necesario el mecanismo de generación de llaves.
- `javax.crypto.SecretKeyFactory`: Una vez creado el PBEKeySpec que contiene la contraseña y, posiblemente, los demás parámetros de generación de llaves, es necesario tener un mecanismo para convertir la contraseña en un objeto que sea compatible con la suite del provedor criptográfico. La clase `SecretKeyFactory` nos sirve para este propósito.

5.2.4. Autoprueba del sistema

Una función de autoprueba debe de realizarse para verificar que los componentes de un modulo esten funcionando correctamente. Debido a que en nuestra aplicación de facturación electrónica maneja datos sensitivos (llave privada, certificados, cartera de clientes, folios) es necesario verificar que estos no hayan sufrido algún tipo de alteración antes de que el sistema de facturación electrónica pueda usarse. El tipo de autoprueba que se implementó en este modulo fue del tipo pre-operacional. Una autoprueba pre-operacional es aquella que se realiza cada vez que el modulo o sistema es encendido o instanciado (despues de ser apagado, reiniciado, después de alguna falla, etc).

En nuestro caso la autoprueba del sistema se realizará al iniciar la aplicación verificando todos los datos sensitivos de la aplicación, en caso de que nuestra aplicación detecte que alguno de los datos sensitivos ha sido alterado, esta tiene la capacidad de pasar a un modo no operacional mostrando un mensaje de error como se muestra en la figura 5.5.

Figura 5.5: Falla de autoprueba del sistema

Para realizar esta función de autoprueba pre-operacional fue necesario la utilización de un componente que pudiera verificar la integridad de los datos, para este propósito se utilizó la función picadillo `SHA1`. Los digestos del conjunto de datos sensitivos ha sido guardado en una pequeña base de datos y cada vez que se realiza esta prueba se verifican los datos de los digestos calculados con los campos de digestos guardados en la base de datos.

5.2.5. Firma electrónica

Una vez generado el documento `XML` con los parámetros de la factura electrónica de acuerdo a Anexo 20 sección C y se ha creado la cadena original de acuerdo a lo que especifica el Anexo 20 sección D, es necesario crear sello digital. El sello digital no es más que la firma digital de la cadena original, este fue creado de acuerdo a los siguientes parámetros [2]:

1. Se aplica el método de digestión `SHA-1` sobre la cadena original a sellar incluyendo los nodos Complementarios. Este procedimiento genera una salida de 160 bits (20 bytes) para todo mensaje. La posibilidad de encontrar dos mensajes distintos que produzcan una misma salida es de 1 en 2^{160}, y por lo tanto en esta posibilidad se basa la inalterabilidad del sello, así como su no reutilización. Es de hecho una medida de la integridad del mensaje sellado, pues toda alteración del mismo provocará una digestión totalmente diferente, por lo que no se podrá autentificar el mensaje.

   `SHA-1` no requiere semilla alguna. El algoritmo cambia su estado de bloque en bloque de acuerdo a la entrada previa.

2. Con la llave privada correspondiente al certificado digital del emisor del mensaje y del sello digital, firmar el digesto del mensaje obtenido en el paso 1 utilizando para ello el algoritmo de cifrado RSA.

3. El resultado será una cadena binaria que no necesariamente consta de caracteres imprimibles, por lo que deberá traducirse a una cadena que sí conste solamente de tales caracteres. Para ello se utilizará el modo de expresión de secuencias de bytes denominado "Base 64", que consiste en la asociación de cada 6 bits de la secuencia a un elemento de un alfabeto que consta de 64 caracteres imprimibles. Puesto que con 6 bits se pueden expresar los números del 0 al 63, si a cada uno de estos valores se le asocia un elemento del alfabeto se garantiza que todo byte de la secuencia original puede ser mapeado a un elemento del alfabeto Base 64, y los dos bits restantes formarán parte del siguiente elemento a mapear.

Por tanto y de acuerdo a lo anterior los algoritmos para generar la llave privada serán los siguientes:

- `SHA-1` y firma con el algoritmo `RSA`: La plataforma Android nos ofrece varias clases para la realización de la firma electrónica con el algoritmo RSA y SHA-1 dentro de los paquetes que nos ayudan a realizar esta tarea se encuentran:

  - `import java.security.Signature`
  - `java.security.interfaces.RSAPrivateKey`
  - `java.security.spec.PKCS8EncodedKeySpec`

A continuación se muestra un pequeño fragmento de código de como se realizó la firma electrónica utilizando los paquetes anteriormente mencionados:

```
PKCS8EncodedKeySpec privSpec = new PKCS8EncodedKeySpec(
    privKeyBytes);
      = (RSAPrivateKey) keyFactory.generatePrivate(
          privSpec);

 // Compute signature
Signature instance = Signature.getInstance("SHA1withRSA",
    "BC");
instance.initSign(privKey);
instance.update(messageToByte);
sigBytes = instance.sign();
```

5.2.6. Bloqueo de la aplicación

La aplicación necesita ser bloqueada cada cierto tiempo para que personas no autorizadas puedan realizar acciones malitencionadas dentro de nuestra aplicación. La aplicación se bloqueará cada 5 minutos una vez que el dispositivo entre en un estado donde no se realicé ninguna acción por parte del usuario (*idle*), mostrando una pantalla que nos pedirá el nombre de usuario y contraseña como se muestra en la figura 5.6.

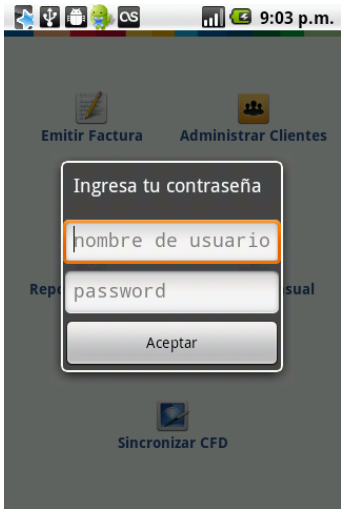


Figura 5.6: Bloqueo de la aplicación

Cada vez que la aplicación se bloquea se borran datos sensitivos como la contraseña de usuario y la llave simétrica con la que se cifran la cartera de clientes y llave privada; esto con el propósito de evitar ataques de volcado de memoria. Una vez que se ingresan los datos de autenticación y sean válidados por la aplicación se volverá a instanciar el cifrador basado en cotraseña.

Para la implementación de de este componente se creo un servicio en Android para capturar los `Intents` generados por la alarma. Se establece un estado global de la instancia del servicio para indicar que la aplicación debe de bloquearse. Para el control de alarma se utilizó la clase `AlarmManager` para programar y cancelar la acción de tiempo de espera como se muestra en el siguiente fragmento de código:

```
ctx.startService(new Intent(ctx, TimeoutService.
    class));
long triggerTime = System.currentTimeMillis() +
    DEFAULT_TIMEOUT;
AlarmManager am = (AlarmManager) ctx.
    getSystemService(Context.ALARM_SERVICE);
am.set(AlarmManager.RTC, triggerTime, buildIntent(
    ctx));
```

El control de la alarma para el bloqueo de la aplicación se controla en los eventos `onPause()` y `onResume()` del ciclo de vida de una aplicación de Android. En el evento `onPause()` se verifica si la aplicación ha estado ocioso si es así se lanza el servicio de alarma, si ocurre alguna interrupción se reinicia la la alarma en el evento `onResume()`

```
@Override
protected void onPause() {
    super.onPause();
```

```
        Timeout.start(this);
    }

    @Override
    protected void onResume() {
        super.onResume();

        Timeout.cancel(this);
        checkShutdown();
    }
```

En la figura 5.7 se describe el diagrama de estados el proceso de bloqueo de la aplicación.

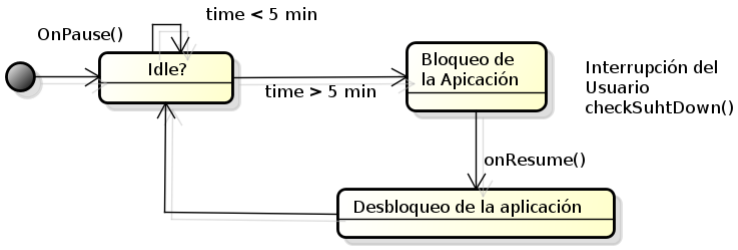


Figura 5.7: Diagrama de estado: Bloqueo de la aplicación

5.3. Pruebas de seguridad

Las pruebas de seguridad es la fase en la cuál el software se somete a un conjunto de pruebas para demostrar que este es seguro. El principal objetivo de las pruebas de seguridad es asegurar que el software que se someta a dichas pruebas sea robusto y continué funcionando de una manera aceptable incluso en la presencia de un ataque mal intencionado. Para fines de este trabajo se realizaron las siguientes pruebas de seguridad:

- **Pruebas funcionales**
- **Pruebas de penetración**

5.3.1. Pruebas funcionales

El objetivo principal de las pruebas funcionales es demostrar que todas las funciones planeadas para proteger el software de ataques estén funcionando correctamente.

Las pruebas se realizaron sobre nuestros componentes de seguridad. Se realizaron tanto pruebas unitarias como pruebas de integración sobre los siguientes componentes:

- **Pruebas de comunicación** : Se verificó que la información entre el dispositivo móvil y el servidor se estuvieran mandando correctamente.
- **Pruebas de soporte criptográfico**: Se verificó que los componentes que proporcionan algún servicio criptográfico funcionaran adecuadamente (modulo de firma digital, modulo de contenedor seguro).
- **Pruebas de autenticación**: Se verificó que los componentes de autenticación permitieran solo acceso a usuarios autorizados.
- **Pruebas de canales seguro**: Se verificó que la información efectivamente se estuviera mandando por un canal confiable.
- **Pruebas de protección de datos**: Se verificó que la información sensitiva efectivamente estuviera protegida.

5.3.2. Pruebas de penetración

Las pruebas de penetración nos sirven para evaluar como un adversario atacará el sistema. En otros términos las pruebas de penetración se refieren a probar el software de una aplicación de software o sistema computacional e intentar comprometer su seguridad. Para realizar las pruebas de penetración existen distintas herramientas las cuáles contienen un conjunto de programas que intentán aprovechar de alguna debilidad del sistema (exploit).

El estado del arte en herramientas de penetración para plataformas móviles es muy pobre. Sin embargo la plataforma Android nos ofrece un conjunto de herramientas que nos puede ayudar de cierta forma a realizar esta tarea a pesar de que estas herramientas no estén destinadas a realizar pruebas de seguridad.

Android Debug Bridge(ADB)

ADB es una herramienta de línea de comandos que permite comunicarse con una instancia del emulador o un dispositivo Android. Es un programa cliente-servidor, que incluye los siguientes tres componentes:

- Un cliente que se ejecuta en el equipo de desarrollo. Se puede mandar a llamar una consola shell desde el emulador o dispositivo con Android.
- Un servidor que se ejecuta como un proceso en segundo plano en el equipo de desarrollo. El servidor gestiona la comunicación entre el cliente y el demonio adb.

- Un demonio, que se ejecuta como un proceso en segundo plano en cada emulador o de instancia del dispositivo.

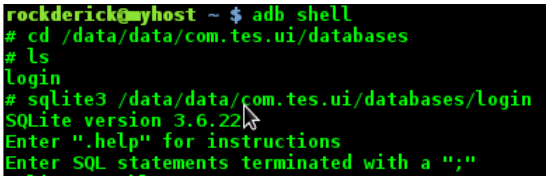
Usando esta herramienta es posible usar los comandos que nos proporciona shell de comandos y así poder extraer información sensitiva de nuestro dispositivo. Bajo la utilización de estas herramientas se plantearon los siguientes escenarios:

Pruebas de penetración en base de datos

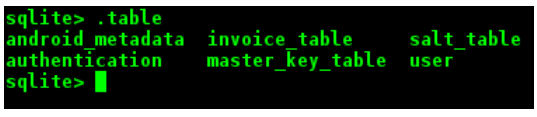
A través de la utilización del shell de comando de ADB es posible ejecutar el comando `sqlite3` para hacer consultas creadas por aplicaciones Android y que esten almacenadas en la memoria del dispositivo. En nuestro caso es posible que un adversario quiera robar la información de contraseñas o llaves que almacenemos en esta base datos. Para realizar este ataque el adversario realizaría lo siguiente:

1. Explorar la bases de datos de las aplicación de facturación electrónica instalada en el dispositivo y ejecutar el comando de `sqlite3`

```
adb -s <nombre del dispositivo> shell
ls /data/data/<aplicacion>/databases/
sqlite3 /data/data/<aplicacion>/databases/<nombredelabasedetaos.db>
```



2. Ejecutar el comando .table para ver la estructura de tablas que tiene la aplicación de facturación electrónica.



Como se puede ver es posible ver nuestro esquema de la base datos. Por ejemplo el adversario podría ver nuestra tabla de control de acceso o nuestra llave del contenedor seguro de nuestra aplicación. Tal como se muestra en la figura 5.7 y la figura 5.8.

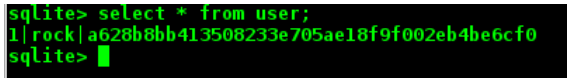


Figura 5.8: Visualización de nuestra contraseña de usuario

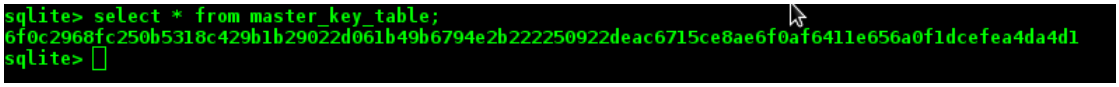


Figura 5.9: Visualización de nuestra llave del contenedor seguro

Como se puede ver este ataque no vulnera nuestro sistema debido a las medidas tomadas en nuestro diseño e implementación del mismo. Sin embargo, si no se hubieran tomado en cuenta estas medidas, el ataque resultaría efectivo y se caería en un fallo grave de seguridad para nuestra aplicación.

Pruebas de penetración de red

Esta prueba se realizó para probar los datos entre nuestro dispositivo móvil y el servidor de respaldo. Para esto es necesario la utilización de un *sniffer* que nos permita capturar y explorar el tráfico de nuestra aplicación. Para realizar esta prueba se utilizará la herramienta wireshark.

En este ataque se supone que existe un adversario escuchando el canal de comunicaciones para tratar de ver la información que pasa entre nuestra aplicación y el servidor de respaldo. Para realizar este ataque el adversario realizaría lo siguiente:

1. Introducir tanto la ip fuente (dispositivo móvil) como la ip destino (servidor de respaldo) en el filtro de la herramienta *wireshark*. La ip fuente y la ip destino pueden ser obtenidas fácilmente por medio de una captura global del tráfico con la herramienta *wireshark*.

```
ip.src==<ip-fuente> and ip.dst==<ip-destino>
```

2. Aplicar la búsqueda de paquetes con este filtro. En nuestro aplicación al realizar la búsqueda con estos parámetros se obtuvieron los resultados de la figura 5.10.

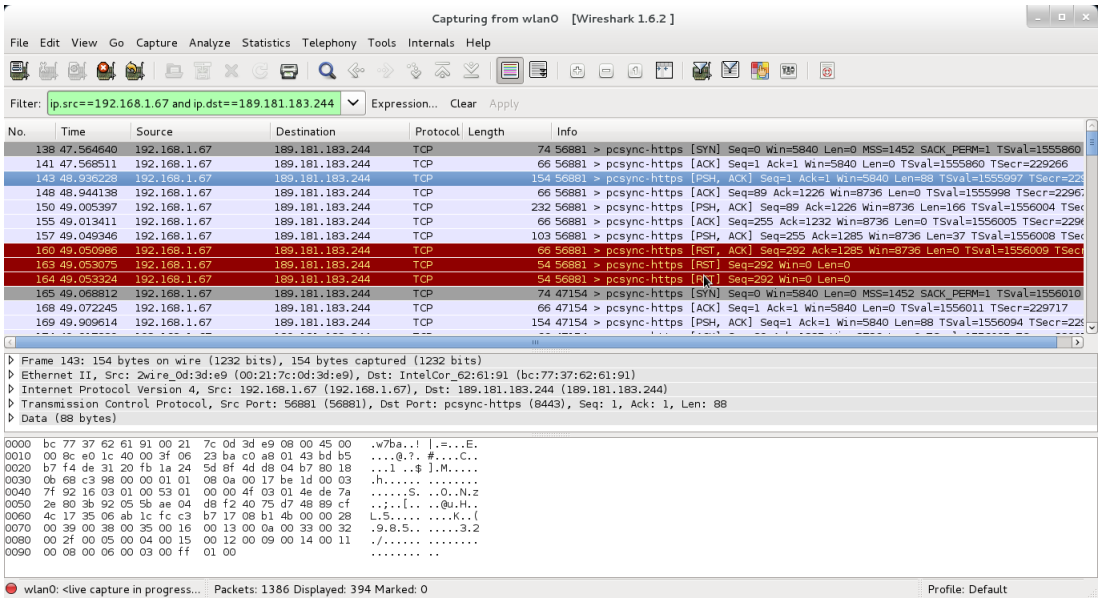


Figura 5.10: Busqueda de peticiones https con wireshark

Como podemos ver debido a que decidimos enviar nuestros parámetros sensitivos por un canal seguro SSL, el adversario no podrá obtener ninguna información relevante de nuestra aplicación. Los datos se estarán enviando por medio del protocolo HTTPS a través del servicio `pcsync-https` que es utilizado por el servidor Apache-Tomcat. Por otra lado, se configuró nuestra aplicación para que se pudieran enviar datos a través del protocolo HTTP para mandar los entre el dispositivo móvil y el servidor. Utilizando los mismos parámetros de búsqueda de paquetes se obtuvo el resultado de la figura 5.11.

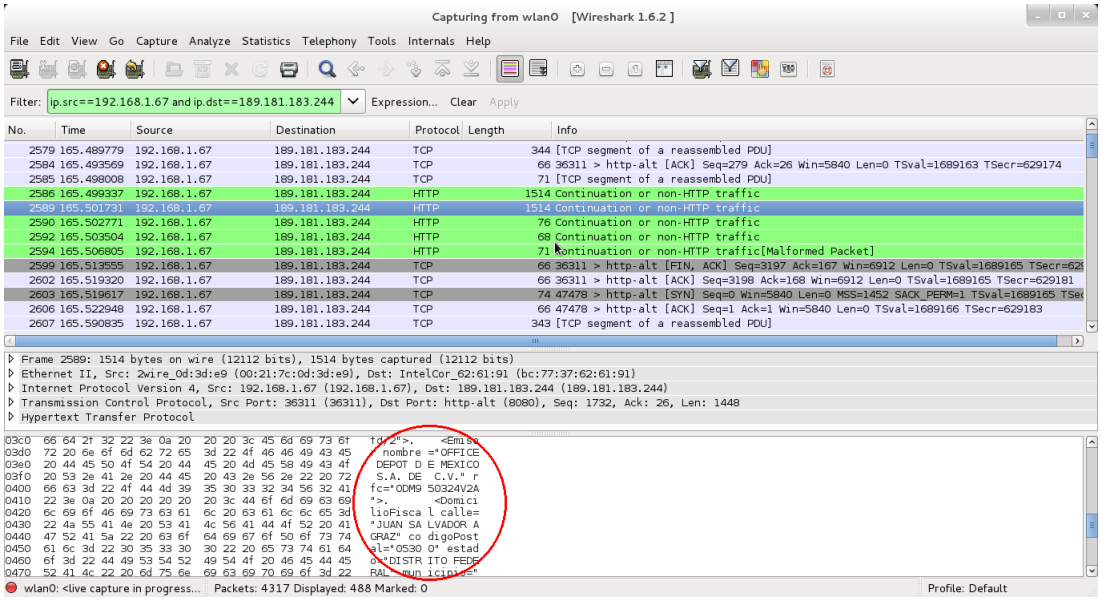


Figura 5.11: Busqueda de peticiones http con wireshark

Como podemos ver, si los datos no son enviados sobre un canal protegido un adversario podría obtener información sobre los parámetros sensitivos de seguridad de nuestra aplicación.

Capítulo 6

Conclusiones y trabajo a futuro

En este capítulo se resumen todas las ideas principales y los resultados obtenidos en la realización del diseño e implementación de un sistema de facturación electrónica. Se mencionaran las principales actividades y aportaciones de este trabajo así como las posibles mejoras que se podrían realizar en el diseño e implementación en el desarrollo seguro de sistemas de facturación electrónica.

6.1. Conclusiones

El principal objetivo de este trabajo de tesis es mostrar un enfoque sistemático para el desarrollo de una aplicación de facturación electrónica para dispositivos móviles incluyendo al ciclo de vida convencional de desarrollo de software el uso de herramientas y metodologías de desarrollo seguro de aplicaciones. Estas herramientas están destinadas a brindar servicios de seguridad tanto en etapas tempranas como en etapas de posteriores del ciclo de vida de desarrollo. Introducir al ciclo de vida convencional el uso de estas herramientas puede mejorar sustancialmente el diseño y la implementación de las aplicaciones. En este trabajo se realizó el análisis, diseño e implementación de una aplicación de facturación electrónica obteniendo como principales resultados los siguientes:

- Se integró a la fase de requerimientos de software la inclusión de la metodología SQUARE la cúal nos sirvió para la obtención y priorización de los requerimientos de seguridad. Esta fase es de suma importancia en el desarrollo si es que se quiere construir un software seguro.
- Se realizó el estudio del análisis y modelado de amenazas através del uso de herramientas como lo son los arboles de ataque y los casos de abuso realizando una estimación de los riesgos de nuestra aplicación. Estas herramientas nos permitieron tener una visión de los objetivos de seguridad para que fueran introducidos en el ciclo de vida de software mostrando los riesgos y debilidades

de este tipo de aplicaciones para posteriormente integrar contra-medidas para mitigar estos.

- Para proveer una plataforma segura de facturación electrónica móvil se realizó el diseño de una arquitectura de software escalable incorporando una capa que se enfoca a prestar ciertos servicios de seguridad a nuestro sistema. El modelado de los componentes de seguridad se hizo utilizando la herramienta UMLSec. El diseño de estos componente podría servir de ayuda para el desarrollo de otro tipo de aplicaciones móviles o de otro tipo aplicando los mismos métodos de protección.
- Se desarrolló una implementación de un sistema de facturación electrónica que incorpora el modelo de seguridad arquitectural propuesto.
- Se incorporó el uso de prácticas de codificación segura las cuales pueden mejorar la calidad del producto.
- A pesar de que las opciones de herramientas para realizar pruebas de penetración son pocas para la plataformas móviles, se presentó una propuesta básica de como se podrían realizar este tipo de pruebas utilizando la plataforma Android.

6.2. Trabajo a futuro

Los siguientes puntos se proponen como trabajo a futuro de este trabajo:

- Incorporar al actual sistema implementado el nuevo esquema de facturación electrónica que entró en vigor en el 2011, el cuál hace incorporación de un timbre digital la cuál va impedir ataques de falsificación de facturas ya que la creación de este este timbre digital es creado por un proveedor autorizado del SAT.
- Implementar este desarrollo en otras plataformas con recursos similares a Android (IOS, Symbian-OS, Meego) y experimentar la implementación en plataformas con más bajos recursos que cuenten con la plataforma J2ME.
- Implementar los controles de seguridad en plataformas de facturación electrónica de mayor escala como podrían ser plataformas de escritorio o plataformas web y realizar el análisis de seguridad para ver que componentes son necesarios en este tipo de sistemas

Apéndice A

Definiciones de seguridad

Activo: Se trata de un recurso de valor que se debe proteger. Pueden ser activos intangibles como la reputación de su empresa, o tangibles como la información que se procesa o los datos de un cliente. También se podría considerar un activo un recurso que utilizado de forma indebida facilite el acceso no autorizado o provoque la no disponibilidad de otro activo.[29]

Algoritmo criptográfico: Un procedimiendo computacionalmente bien definido que toma variables de entrada, el cuál incluye llaves criptográficas y produce una salida [23].

Amenaza: Un evento de posible ocurrencia que podría dañar o comprometer un activo o un objetivo estratégico.[29]

Ataque: Una acción que se sirve de una o varias vulnerabilidades para materializar una amenaza.[29]

Autenticación: El proceso de determinar si alguien o algo es en verdad lo que dice ser. [27]

Ataque de denegación de servicio: Una forma de ataque a una computadora o grupo de computadoras enviando millones de peticiones cada segundo, ralentizando la red. [27]

Cadena original: Se entiende como cadena original, a la secuencia de datos formada con la información contenida dentro de la factura electrónica, establecida en el Rubro C del Anexo 20 de la Resolución Miscelánea Fiscal [5].

Certificado de sello digital: Documento electrónico, mensaje de datos u otro registro que asocia una llave pública con la identidad de su propietario, confirmando el vínculo entre éste y los datos de creación de una firma electrónica avanzada o de un sello digital [5].

Cliente o receptor: Persona que recibirá las fácturas electrónicas [5].

Codigo de Autenticación de Mensajes: Una checksum de los datos que usa una llave simétrica para detectar tanto modificaciones accidentales como intencionales en los datos [23].

Comprobante Fiscal Digital: Es un mecanismo alternativo de comprobación de ingresos, egresos y propiedad de mercancÃas en traslado por medios electrónicos. Maneja estándares de seguridad internacionalmente reconocidos, que garantizan que el comprobante es auténtico, íntegro, único y que será aceptado igual que el comprobante fiscal impreso [5].

Contraseña: Una cadena de caracteres(números, letras y otros simbolos) usados para autenticar una identidad o verificar autorización de acceso [23].

Confidencialidad: Es la propiedad de que la información sensitiva no este disponible o se divulge a individuos, entidades o procesos no autorizados [23].

Firma Digital: Es el resultado de una transformación criptográfica, la cuál de ser implementada correctamente provee los servicios de autencicación de origen, integridad de datos y no repudio del firmante [23].

Fuerza bruta: Una técnia de criptoanálisis o otro tipo de método de ataque que involucra un procedimiento exhaustivo de todas las posiblidades existentes [27].

Funcion picadillo o hash: Una función computacionalmente eficiente que proyecta cadenas de longitud arbitraria a cadenas de longitud fija, la cuál es computacionalmente imposible encontrar dos valores con el mismo digesto [23].

Impacto: Efecto negativo de un ataque sobre un activo o víctima por algún atacante [29].

Integridad: La propiedad de que información sensitiva no sea modificada o borrada de manera no autorizada sin existir alguna detección. [23]

Llave cifrada: Una llave criptográfica que ha sido cifrada una función segura per-

mitida o aprobada [23].

Llave pública: Una llave criptográfica usada con un algoritmo criptográfico público, y que esta unicamente asociado con una entidad [23].

Llave privada: Llave criptográfica, usada con un algoritmo criptográfico, que esta únicamente asociada con una entidad y esta no se hace pública [23].

Llave simétrica o secreta: Una llave criptográfica, usada con un algoritmo criptográfico de llave secreta y que esta unicamente asociada con una o más entidades y que no debe de hacerse pública [23].

Información o datos sensitivos: datos que deben de ser protegidos [23].

Parametro crítico de seguridad (CSP): Cualquier información secreta (e.g., llave privada o secreta, llave compartida y datos de autenticación tales como contraseñas) la cuál su divulgación o modificación pueda comprometer la seguridad de el modulo criptográfico [23].

Parametro de seguridad crítico público: Cualquier parametro público de seguridad el cuál su modificación pueda compromenter la seguridad del modulo de seguridad o criptográfico [23].

Parametros sensitivos de seguridad: Parametros de seguridad criticos y parametros públicos de seguridad. [23].

Penetración: Intrusión o acceso no autorizado en el sistema. [27]

Privacidad: La condición de ser aislado de la presencia o vista de otros. [27]

Sello digital: Mecanismo que permite acreditar la autoría de los comprobantes fiscales digitales que se emitan, de esta manera sus clientes sabrán la identidad del autor del comprobante fiscal digital [5].

Usuario: Persona que hace uso de el sistema de facturación electrónica. [5]

Vulnerabilidad: Una vulnerabilidad es un punto débil que de explotarse con éxito puede llegar a originar la consecución de una amenaza [29].

Apéndice B

Identificación de metas de seguridad

B.1. Confidencialidad

El objetivo de confidencialidad para la aplicación de facturación electrónica es lograr que la información se encuentre protegida ante usuarios no autorizados. La aplicación debe de contar con controles de seguridad para no divulgar información de los siguientes activos: llave privada, folios, certificado digital, cartera de clientes, factura emitida y entregada (por entregar), copia local de facturas emitidas, copia respaldo de facturas emitidas y reporte mensual. Estos controles de seguridad deben de existir tanto en el dispositivo móvil, como en el servidor de respaldo. Así mismo se deben de contar con políticas para mantener la no divulgación, y expiración de las contraseñas.

B.2. Disponibilidad.

La aplicación de facturación electrónica existen ciertas funciones que son indispensables para el funcionamiento del sistema. Funciones como la emisión de facturas, sincronización de datos o de facturas son críticas para el sistema por lo tanto deben de estar siempre disponibles para el sistema. Adicionalmente de guardar facturas electrónicas y la cartera de clientes de forma local se hará un respaldo hacia un servidor remoto para respaldar estos datos. El respaldo se hará por los usuarios al final de día.

B.3. Integridad de datos

Los datos como la llave privada, folios, certificado digital, cartera de clientes, factura emitida y entregada (por entregar), copia local de facturas emitidas, copia respaldo de facturas emitidas, reporte mensual no deben deben de ser alterados. Antes de utilizar el sistema de facturación electrónica se debe de realizar una función de autoprueba que cheque la integridad tanto de la aplicación como de los datos anteriormente mencionados, si la aplicación ha sufrido alguna alteración en los datos el sistema tendrá la capacidad de alertar al usuario.

B.4. Seguimiento

Se debe de contar con un seguimiento de todas las acciones que usuario del sistema ocupe através de una bitácora. Acciones como la emisión de facturas, sincronización, altas, bajas y modificaciones en la cartera de clientes tendrán que ser registradas para poder ser consultadas por el administrador.

Apéndice C

Casos de uso

Identificador	Crear una nueva cuenta principal del sistema	
Nombre	Acción “Crea una nueva cuenta de usuario ”	
Descripción	El usuario crea un nueva cuenta.	
Precondición	-El software acaba de ser instalado y es la primera vez que sera usado. Los folios, llaves y certificados están previamente cargados en el dispositivo	
Actores principales	Administrador.	
Secuencia normal	Paso	Acción
	1	El usuario selecciona la opción registrarse.
	2	Se le pide al usuario que inserte un nombre de usuario y contraseña nueva. La contraseña que se introduzca debe de estar basado en un estándar deberá de contener letras, números, caracteres especiales.
	3	Se crea contenedor seguro con la cuenta del usuario que contendrá llave privada, folios y certificados, cartera de clientes
Postcondición	El usuario solicita entrar al sistema a través de el control acceso.El sistema pasa a un nuevo estado, donde todos los componentes del sistema están debidamente inicializados(certificados, folios, llave privada, contenedor seguro). El software se encuentra en estado operativo.	
Frecuencia esperada	Una vez en el primer uso del sistema	
Importancia	Alta, indispensable para inicializar el sistema por primera vez y cargar llave privada, certificados y folios.	

Cuadro C.1: Caso de Uso CU-01

Identificador	CONTROL DE ACCESO	
Nombre	Acción “acceso al sistema por medio de el control de acceso principal”	
Descripción	El usuario pueden tener acceso al sistema.	
Precondición	Antes de realizar el acceso al sistema se realizara una función de autoprueba, para ver que todos los componentes del software no hayan sido alterados. Este proceso se realiza en la carga de la aplicación.El usuario ya debe de estar dado de alta en el sistema.	
Actores principales	Usuario.	
Secuencia normal	Paso	Acción
	1	El usuario para tener acceso al sistema debe de ingresar su nombre de usuario y contraseña, dando la instrucción de acceso al sistema
	1a	Si el id de usuario y contraseña son correctos se podrá acceder exitosamente
	1b	Si el id de usuario y/o contraseña son incorrectos se procede a la excepción autherr.
Postcondición	Ingresa exitosamente al sistema de facturación	
Excepciones	Excepción auterr	
	Paso	Acción
	1	Si el nombre de usuario y/o contraseña son erróneos mostrara un mensaje de error
Frecuencia esperada	Una vez por cada usuario por cada intento de acceder al sistema	
Importancia	Alta, Indispensable para controlar el acceso al sistema	

Cuadro C.2: Caso de Uso CU-02

Identificador	Generació de factura	
Nombre	Acción “Generar Factura”	
Descripción	El usuario podrán generar una factura electrónica a un cliente	
Precondición	El usuario debe de estar previamente registrado en el sistema.	
Actores principales	Usuario.	
Secuencia normal	Paso	Acción
	1	El usuario selecciona la opción “Crear Factura” dentro el menú principal del sistema.
	2	El usuario selecciona el cliente de una lista de clientes registrados(contribuyente), en caso contrario se podrá ingresar un nuevo usuario.
	3	Al seleccionar un cliente el sistema, mostrará los datos del cliente tales como: Nombre de la empresa, RFC, Domicilio Fiscal, Datos y Cantidad del producto.
	4	El usuario debe de de insertar los datos del producto(s) por el cuál se desea realizar la factura. Se deben de ingresar todos los datos del producto para poder realizar la factura. Los datos que se deben de insertar son los siguientes: Cantidad, Descripción, Si genera IVA, Precio Unitario. Este proceso se realiza por cada producto o servicio del cuál se requiera la factura.
	4.a	El sistema calcula los siguientes datos: Importe, Subtotal, Precio con IVA, Retención de IVA(Si aplica), Retención de ISR(Si aplica), TOTAL
	5	Una vez insertados todos los datos se muestran para confirmación y se puede proceder a realizar la factura electrónica, seleccionando el botón crear factura.
	6	Al momento de presionar el botón, se realizaran los siguientes pasos:
	6a	Se genera un archivo XML, con los parámetros que apliquen para dicha factura. Estos parámetros serán los que que se especifican en el Anexo 20 sección C.

Secuencia normal	Paso	Acción
	6b	Se procede a generar el sello digital de acuerdo al Anexo 20 sección D. Para realizar este proceso es necesario cargar la llave privada en la memoria , al finalizar el sello digital, la llave privada debe de ser borrada de la memoria de manera segura del dispositivo. Los pasos a seguir para realizar el sello digital son los siguientes: 1) Se genera la cadena original, 2) Se recupera la llave privada desde el repositorio seguro, 3) Con la llave privada creada en memoria se realiza la firma digital de la cadena original, 4) Se borra de memoria la llave privada, 5) Se anexa al documento xml el sello digital.Si existe algún problema con la llave o la firma digital se procederá a la excepción **signerr.**
	6c	Una vez generado el archivo XML y el sello digital, se procederá a enviar este archivo vía un canal seguro hacia el cliente(contribuyente).Se guarda las facturas en el repositorio seguro.
	6d	Si se generó y se envió correctamente el archivo XML, el sistema mostrara una notificación de éxito En caso contrario se procederá a la excepción **generr** o **generr2** según sea el caso**.**
	7	Se muestra la opción de regresar al menú principal.
Postcondición	La facturas son guardadas en formato XML(solo se mantendrán en este formato y no se tendrá una base de datos relacional para su almacenamiento).	
Excepciones	Excepción signerr	
	Paso	Acción
	1	Si en el momento de hacer la firma, la llave privada no se encuentra dentro de la memoria SD o si existe un error de entrada/salida al leer la llave privada se realizan los siguientes pasos:
	1.a	Si el sistema notifica un error al no encontrar la llave privada en la memoria SD el sistema notificará al usuario y cerrando la aplicación
	1.b	Si el sistema notifica un error de entrada/salida al leer la llave privada el sistema notifica al usuario y cerrara la aplicación.

	Excepción generr	
	Paso	Acción
	1	La generación del archivo tuvo un problema al momento de escribirse a disco o no se realiza correctamente el sello digital.
	2	El archivo XML no se manda correctamente al contribuyente.
	3	Se lanza una notificación al usuario para que vuelva a realizar la factura
	Excepción generr2	
	1	Existe un problema de conexión y no se puede enviar el archivo XML
	2	El sistema guarda el archivo y lo marca como pendiente para poder mandarlo posteriormente y recuperarse de dicha excepción.
Frecuencia esperada	Cada vez que el usuario genere una factura	
Importancia	Alta, indispensable para la generación de comprobantes fiscales digitales.	

Cuadro C.5: Caso de Uso CU-03

Identificador	Alta de Cliente	
Nombre	Acción “Dar de Alta un Nuevo Cliente del Sistema”	
Descripción	El usuario puede dar de alta un cliente nuevo al cuál se le va a facturar.	
Precondición	El usuario debe de estar previamente registrado en el sistema.	
Actores principales	Usuario.	
Secuencia normal	Paso	Acción
	1	El usuario selecciona la opción “Alta de Clientes” dentro el menú principal del sistema.
	2	El cliente ingresa los siguientes datos al sistema: Razón Social, Correo Electrónico, Dirección, RFC. Si alguno de los datos son inválidos se procederá a la excepción **datosfiscerr.**
	3	Los datos son almacenados en la base de datos local del sistema. Si no existe espacio en disco duro del dispositivo se procederá a la excepción **nodiskspaceerr**
	4	Se notifica al usuario que el usuario ha sido creado exitosamente
	5	Se muestra la opción de regresar al menu principal.
Postcondición	Los datos del usuario deberán ser respaldados en un servidor remoto(sincronización de datos).	
Excepciones	Excepción datosfiserr	
	Paso	Acción
	1	Si alguno de los datos del paso 2 son incorrectos el sistema mostrara un mensaje de error.
	2	Los datos deberán de ingresarse de nuevo.
	Excepción nodiskspaceerr	
	Paso	Acción
	1	Si no hay espacio en disco duro para hacer el almacenamiento de un nuevo cliente el sistema el sistema notifica al usuario, indicándole que borre archivos no esenciales del dispositivo.
Frecuencia esperada	Cada vez que el usuario quiera dar de alta un nuevo cliente	
Importancia	Media. Indispensable para dar de alta nuevos clientes.	

Cuadro C.6: Caso de Uso CU-04

Identificador	Baja de cliente	
Nombre	Acción “Dar de baja un Cliente del Sistema”	
Descripción	El usuario puede dar de baja un cliente del sistema previamente registrado.	
Precondición	El usuario debe de estar previamente registrado en el sistema.Se debe tener por lo menos registrado un cliente en el sistema.	
Actores principales	Usuario.	
Secuencia normal	Paso	Acción
	1	El usuario selecciona la opción “Administración de Clientes” dentro el menú principal del sistema.
	2	El usuario selecciona la opción “Dar de baja un cliente” dentro del menú Administración de Clientes.
	3	El usuario realiza una búsqueda por algún de los siguientes criterios. Razón Social, Correo Electrónico o RFC.
	4	Se muestran una lista de los clientes de acuerdo a las coincidencias del criterio de selección.
	5	Se selecciona al cliente que se quiere borrar.
	6	Se borran los datos del cliente de la base de datos local. Si existe algún error en la eliminación de la base de datos se procederá a la excepción **databaseerr.**
	7	Se muestra la opción de regresar al menú principal.
Postcondición	Los cambios realizados al eliminar el cliente se verán reflejados al sincronizar los datos en el servidor remoto.	
Excepciones	Excepción databaseerr.	
	Paso	Acción
	1	Existe algún problema en la eliminación de un usuario, se le notifica al usuario error. Se cancela la operación de borrado.
Frecuencia esperada	Cada vez que el usuario quiera eliminar un cliente.	
Importancia	Baja solo es indispensable para eliminar clientes del sistema.	

Cuadro C.7: Caso de Uso CU-05

Identificador	Cambios de datos de cliente	
Nombre	Acción “Modificación de un Cliente del Sistema”	
Descripción	El usuario puede modificar un usuario del sistema previamente registrado.	
Precondición	El usuario debe de haber accedido por el control de acceso principal del sistema. Se debe tener por lo menos registrado un cliente en el sistema.	
Actores principales	Usuario.	
Secuencia normal	Paso	Acción
	1	El usuario selecciona la opción “Administración de Clientes” dentro el menú principal del sistema.
	2	El usuario selecciona la opción “Modificar datos de cliente” dentro del menú Administración de Clientes.
	3	El usuario realiza una búsqueda por los siguientes criterios: Razón Social, Correo Electrónico3, Dirección, RFC.
	5	El usuario modifica al menos uno de los siguientes datos: Razón Social, Correo Electrónico3, Dirección, RFC.Si los dato(s) ingresados son llenados incorrectamente se procederá a la excepción **datosfiserr** .
	6	Se pide confirmación explicita del usuario mediante las ordenes actualizar o cancelar.
	7	La modificación de los datos es guardada en la base de datos local. Si no existe espacio en disco duro del dispositivo se procederá a la excepción **nodiskspaceerr**
	7	Se notifica al usuario que el usuario ha sido modificado exitosamente
	8	Se muestra la opción de regresar al menu principal.
Postcondición	Los datos del usuario modificados serán respaldados en un servidor remoto(sincronización)	
Excepciones	Excepción datosfiserr Excepción nodiskspaceerr	
	Paso	Acción
	1	Si no hay espacio en disco duro para hacer el almacenamiento de un nuevo cliente el sistema alerta al usuario, indicándole que borre archivos no esenciales del dispositivo.
Frecuencia esperada	Cada vez que el usuario quiera modificar información del cliente.	
Importancia	Baja, se puede prescindir de este caso de uso y el sistema seguirá funcionando.	

Cuadro C.8: Caso de Uso CU-06

Identificador	Generar Reporte Mensual	
Nombre	Acción "Generar Reporte Mensual del SAT."	
Descripción	El usuario selecciona la opción generar el reporte mensual de acuerdo a lo establecido en el anexo 20 sección A.	
Precondición	El sistema realizó la autoprueba para ver que todos los componentes del sistema son correctos. El usuario debe de estar previamente registrado en el sistema.	
Actores principales	Usuario	
Secuencia normal	Paso	Acción
	1	El usuario selecciona la opción generar el reporte mensual.
	2	El usuario selecciona el periodo del reporte mensual del sistema.
	3	El sistema extrae los datos que determina el Anexo 20 sección A de cada CFD dentro de la colección de archivos XML.Si existe algún problema de lectura en algún archivo XML se procederá a la excepción: **xmlioerr.**
	4	Se crea un archivo de texto de acuerdo a lo establecido en el Anexo 20 sección A, que se guarda en la memoria local del dispositivo. Si ocurre un error en la escritura del archivo se procede a la excepción **txtioerr.**
	5	El archivo es guardado en memoria para luego ser enviado al SAT por lo medios que este determine. Si ocurre un error en el almacenamiento se procede a la excepción **saverr.**
	6	Se muestra la opción de regresar al menu principal.
Postcondición	Se guarda el reporte mensual.	
Excepciones	Excepción xmlioerr	
	Paso	Acción
	1	Se lee erróneamente algún CFD's que están en formato XML.
	2	El sistema manda una notificación de error.
	3	El sistema reinicia los pasos del caso de uso de "Generar Reporte Mensual" desde el paso 1
	Excepción txtioerr	
	Paso	Acción
	1	Existe un error en la escritura del reporte mensual
	2	El sistema manda una notificación de error.
	3	El sistema reinicia los pasos del caso de uso "Generar Reporte Mensual" desde el paso 4
	Excepción	saveerr
	1	Existe un error en el almacenamiento del reporte mensual
	2	Se le notifica al usuario que haga de nuevo el reporte mensual
Frecuencia esperada	Cada día primero de cada vez	
Importancia	Alta, es necesario ya que el SAT lo solicita cada mes	

Cuadro C.9: Caso de Uso CU-07

Identificador	Reporte Contable	
Nombre	Acción “Visualizar Reporte Mensual”	
Descripción	El usuario puede ver el Reporte Mensual de todas las facturas emitidas.	
Precondición	El usuario debe de haber ingresado a través del control de acceso de el sistema.	
Actores principales	Usuario.	
Secuencia normal	Paso	Acción
	1	El usuario selecciona la opción “Reporte Contable” dentro el menú principal del sistema.
	2	Se muestra una lista de todas las facturas emitidas.
	3	Se muestra la opción de regresar al menu principal.
Frecuencia esperada	Cada vez que se quiera consultar las facturas emitidas.	
Importancia	Visualización de las facturas emitidas	
Urgencia	Baja, no es indispensable para el funcionamiento del sistema, solo es de carácter informativo para el usuario	

Cuadro C.10: Caso de Uso CU-08

Identificador	Consulta de CFD's	
Nombre	Acción "Consulta de CFD's emitidos"	
Descripción	El usuario puede consultar los CFD's emitidos.	
Precondición	El usuario debe de haber ingresado a través del control de acceso del sistema. Se debe haber emitido al menos una factura.	
Actores principales	Administrador, Usuario.	
Secuencia normal	Paso	Acción
	1	El usuario selecciona la opción "Consulta de CFD's" dentro del menú principal del sistema.
	2	Se selecciona alguno de los siguientes filtros de búsqueda: Emisor, Receptor o Producto.
	3	Se inserta el criterio de búsqueda y se realiza la búsqueda sobre los CFD's almacenados. Si existe un error en la búsqueda en la base de datos se procede a la excepción **databaseerr.**
	4	Se muestra una lista de el/los CFD's presentes en el dispositivo encontrados de acuerdo al criterio de la búsqueda o si no la hay se muestra una notificación de que no existieron coincidencias en la búsqueda.
	5	Se selecciona el CFD deseado.
	6	Se muestra a detalle el CFD emitido.
	7	Se muestra la opción regresar de el menú principal.
Excepciones	Excepción databaseerr.	
	Paso	Acción
	1	Existe algún problema al acceder los datos en la base de datos.
	2	El sistema muestra una notificación para volver a intentar la búsqueda.
Frecuencia esperada	Cada vez que el usuario consulte un CFD.	
Importancia	Baja, no es indispensable para el funcionamiento primordial del sistema sin embargo es útil para poder revisar una bitácora de que facturas se han emitido.	

Cuadro C.11: Caso de Uso CU-09

Identificador	Sincronizar datos	
Nombre	Acción “Se sincronizan los cfd y la cartera de clientes con el servidor para ser respaldados”	
Descripción	El usuario elige la opción de sincronizar los datos, y estos posteriormente serán respaldados.	
Precondición	Debe de existir al menos un cliente en la cartera de clientes o una factura emitida.	
Actores principales	Usuario, Servidor de respaldo.	
Secuencia normal	Paso	Acción
	1	Se selecciona de la pantalla principal la opción “Sincronizar Datos”
	3	Los datos son mandados a un servidor vía un canal seguro. Si existe un error se procede a la excepción **comerr**
	4	El servidor recibe los datos.
	5	Los datos son almacenados en una base de datos.
	6	Se envía una notificación al dispositivo diciendo que los datos han sido respaldados correctamente.
	7	El sistema recibe la notificación y alerta al usuario que se realizo con éxito el respaldo.
	8	Se cambia el estado de los archivos a respaldados.
	9	Se muestra la opción de regresar al menú principal.
Postcondición		
Excepciones	Excepción comerr	
	Paso	Acción
	1	Se determina que no existe comunicación hacia el servidor o que la comunicación ha sido interrumpida.
	2	El sistema muestra una notificación avisando que el sistema de sincronización ha fallado y se le pide al usuario que lo realice posteriormente.
Frecuencia esperada	Cada vez que el administrador quiera respaldar los datos del dispositivo.	
Importancia	Indispensable para que exista un respaldo de los datos del dispositivo.	

Cuadro C.12: Caso de Uso CU-10

Apéndice D

Casos de maluso

Numero:	MC-01	
Nombre:	Genera Factura sin acceso a llave privada(residual)	
Alcance:	Compromete la misión de la aplicación	
Prioridad:	Prioridad Alta	
Ambiente:	Host y Aplicación.	
Anti-Actores:	Usuarios no Autorizados	
Niveles de derecho de acceso:	Nivel Usuario Nivel Administrador	
Puntos de entrada:	Aplicación	
Atributos de seguridad afectados:	Autenticidad de la información,no repudio	
Descripción:	Una vez que se ha generado una factura previa, es posible que la llave privada haya quedado en memoria. Dicha copia podría usarse para generar una factura posteriormente y sin que esto requiera verificar al contenedor de la llave privada.	
Precondiciones:	Se accede a la llave privada para firmar una factura electrónica de forma local en el dispositivo(se requiere que por lo menos se haya generado una firma para poder realizar el ataque). Se accede al proceso gen_factura() y se mantiene en memoria la copia de la llave privada. No se inicializó la variable que contiene la llave privada. No se cerro la sesión de la applicación. Se tenian folios cargados en el dispositivo.	
Suposiciones:	Se supone que este ataque se realiza mediante el robo del dispositivo.	
Post-Condiciones:	Peor Caso	Se genera fáctura válida sin aviso al contribuyente
	Medida de prevención deseada	-Blanquear llave privada -Lock de la aplicación.
	Medida de detección deseada	-Bitacora de la aplicación realizando un corte diario -Cruce de las facturas emitidas contra el SAT.
	Medidia de recuperación deseada.	-Revocar la llave privada. -Litigio.
Perfiles de potenciales anti-actores:	Usuarios sin conocimiento técnico agudo.	

Clientes y amenazas:	Clientes: creación de facturas sin consentimiento del contribuyente.
Casos de uso relacionados:	
Amenazas relacionadas:	Elevación de privilegios, acceso no autorizado a la interfaz,autenticidad de los datos, usurpación de la identidad,repudiación de la información.
Recomendaciones arquitecturales:	-Se debe de realizar un bloqueo de la aplicación cada 10 min si la aplicación no esta en uso.-Cada vez que se realice un factura se de pedir el password de la aplicación o la del acceso a la llave privada.-Crear bitacoras de las operaciones dela aplicación.-Blanqueo de la llave privada cada vez que se ocupe.
Políticas de Recomendación:	-Realizar cortes de caja al día y ver las facturas que se han emitido-El password que se utilice para generar la factura electrónica debe de ser cambiado periódicamente.-Contar con políticas para contraseñas fuertes.-Acciones de usuarios deben de ser revisadas periódicamente.

Cuadro D.2: Caso del Mal Uso MC-01

Numero:	MC-02	
Nombre:	Creación de factura electrónica a través de robo del contenedor de la llave privada	
Alcance:	Compromete el funcionamiento de la aplición	
Prioridad:	Alta.	
Ambiente:	Hardware	
Anti-Actores:	Usuarios no Autorizados	
Niveles de derecho de acceso:	Ninguno	
Puntos de entrada:	Hardware	
Atributos de seguridad afectados:	Auntenticiadad de la información	
Descripción:	Un usuario extrae el contenedor donde se encuentran la factura electrónica y los folios, e intentara realizar una factura electrónica por sus propios medios.	
Precondiciones:	El contenedor es extraído del dispositivo. Se cuenta con una aplicación similar para realizar la aplicación de facturación electrónica. Se tienen folios precargados.	
Suposiciones:		
Post-Condiciones:	Peor Caso	Crea una factura electrónica.
	Medida de prevención deseada	Los datos que esten en el contenedor deben de estar cifrados
	Medida de detección deseada	Bitacora de acciones realizadas en el dispositivo.
	Medidia de recuperación deseada.	-Revocar la llave privada. -Respaldar datos importantes
Perfiles de potenciales anti-actores:	Usuario sin conocimientos técnicos.	
Clientes y amenazas:	Clientes: creación de facturas sin consentimiento del contribuyente.	
Casos de uso relacionados:		
Amenazas relacionadas:	Acceso no autorizado a la interfaz,Usurpación de la identidad,repudiación de la información.	
Recomendaciones arquitecturales:	-Se debe de tener una bitácora de las acciones que se realicen en el dispositivo.-Los datos como los folios, llave privada deben de estar cifrados en el contenerodr.	
Políticas de Recomendación:	-Hacer un cruce de las facturas emitidas contra el SAT y verificar cuales de estas no se han emitido por el contribuyente.	

Cuadro D.3: Caso del Mal Uso MC-02

Numero:	MC-03	
Nombre:	Suplantación de contribuyente al generar facturas a su nombre.	
Alcance:	Compromete el funcionamiento de la aplicación.	
Prioridad:	Alta.	
Ambiente:	Hardware del dispositivo.	
Anti-Actores:	Usuarios no Autorizados	
Niveles de derecho de acceso:	Ninguno	
Puntos de entrada:	Hardware del dispositivo.	
Atributos de seguridad afectados:	Autenticidad de la información, no repudio, integridad de los datos contenidos	
Descripción:	Se extrae el contenedor del dispositivo y se ingresan otros datos de diferente contribuyente con folios válidos.	
Precondiciones:	Se introduce información mediante otro contenedor de datos	
Suposiciones:	Se tiene acceso directo al dispositivo	
Post-Condiciones:	Peor Caso	Se realiza una factura con otros datos.
	Medida de prevención deseada	Verificar los datos del certificado contra los del servidor.
	Medida de detección deseada	-Tener bitácora de las acciones del emisor de la factura.
	Medidia de recuperación deseada.	-Revocación de el certificado. -Notificación al SAT
Perfiles de potenciales anti-actores:	Usuarios sin conocimientos técnicos agudos.	
Clientes y amenazas:	Clientes: creación de facturas sin consentimiento del contribuyente.	
Amenazas relacionadas:	Acceso no autorizado a la interfaz,Usurpación de la identidad,repudiación de la información,modificación de datos	
Recomendaciones arquitecturales:	-Contar con un servidor de base de datos que tenga la información de el contribuyente.-Se debe de tener una bitácora de las acciones que se realicen en el dispositivo.	

Cuadro D.4: Caso del Mal Uso MC-03

Numero:	MC-04	
Nombre:	Creación de una factura electrónica accediendo a la llave privada.	
Alcance:	Compromete el funcionamiento de la aplicación.	
Prioridad:	Alta.	
Ambiente:	Hardware del dispositivo.	
Anti-Actores:	Usuarios no Autorizados	
Niveles de derecho de acceso:	Ninguno	
Puntos de entrada:	Hardware del dispositivo.	
Atributos de seguridad afectados:	Autenticidad de los datos, no repudio	
Descripción:	Se obtiene la llave privada haciendo un volcado de memoria hacia el dispositivo buscando residuos de la llave privada que se ocupo para realizar la llave privada.	
Precondiciones:	1) El atacante obtiene acceso al dispositivo 2) El atacante hace un volcado de memoria para ver si es posible encontrar la llave privada. 3) Se tuvo que haber realizado por lo menos una vez la factura para poder obtener información mediante el volcado de memoria. 4) Se accede mediante otra aplicación que realice facturación electrónica con la llave privada extraída por el volcado de memoria. 5) Se accede al método de generar_factura de dicha aplicación.	
Suposiciones:	Se cuentan con folios del contribuyente al cual se le esta realizando el ataque	
Post-Condiciones:	Peor Caso	El atacante obtiene la llave privada y puede realizar facturas sin consentimiento del contribuyente.
	Medida de prevención deseada	Blanqueo de memoria de la llave privada.
	Medida de detección deseada	
	Medidia de recuperación deseada.	-Revocación de la llave privada.
Perfiles de potenciales anti-actores:	Usuarios sin conocimientos técnicos agudos.	
Clientes y amenazas:	Clientes: creación de facturas sin consentimiento del contribuyente.	
Amenazas relacionadas:	Acceso no autorizado a la interfaz,Usurpación de la identidad,repudiación de la información,revelación de información.	
Recomendaciones arquitecturales:	Blanqueo de memoria	

Cuadro D.5: Caso del Mal Uso MC-04

Numero:	MC-05	
Nombre:	Creación de Factura generando un certificado de sello digital apocrifo.	
Alcance:	Compromete el funcionamiento de la aplicación.	
Prioridad:	Alta	
Ambiente:	Hardaware	
Anti-Actores:	Usuario no autorizado	
Niveles de derecho de acceso:	Nivel Usuario Nivel Administrador	
Puntos de entrada:	Dispositivo	
Atributos de seguridad afectados:	Autenticiadad de los datos,No repudio	
Descripción:	Debido a que el sello digital es realizado con funciones picadillo obseletas(MD-5,SHA-1) es posible que se generen certificados falsos que parezcan válidos.	
Precondiciones:	1.- Se inserta un certificado que parece válido.2.- Se accede al método de gen_factura. 3. Se cuenta con folios en el dispositivo.4.La autoridad certificadora no detecta que el certificado es inválido	
Suposiciones:	Se cuenta con infraestructura necesaria para encontrar colisiones en las funciones picadillo MD-5,SHA-1,SHA-2	
Post-Condiciones:	Peor Caso	Se genera una factura.
	Medida de prevención deseada	Cambiar los algoritmos para realizar funciones picadillo
	Medida de detección deseada	No existe
	Medidia de recuperación deseada.	No existe
Perfiles de potenciales anti-actores:	Usuarios especializados con conocimiento técnicos avanzados	
Clientes y amenazas:	Clientes: creación de facturas sin consentimiento del contribuyente.	
Casos de uso relacionados:		
Amenazas relacionadas:	Usurpación de la identidad	
Recomendaciones arquitecturales:	-Cambiar los estándares propuestos por el SAT(proponer SHA-512 por el momento).	

Cuadro D.6: Caso del Mal Uso MC-05

Numero:	MC-06	
Nombre:	Modificar/eliminar los datos del CFD o datos de la cartera de clientes mediante su transporte hacia el receptor o hacia el servidor del respaldo.	
Alcance:	Compromete el funcionamiento de la aplicación.	
Prioridad:	Baja	
Ambiente:	Internet	
Anti-Actores:	Integridad de la información,Disponibilidad	
Niveles de derecho de acceso:	Nivel usuario o admininstrador	
Puntos de entrada:	Canal de transmisión	
Atributos de seguridad afectados:	Autenticidad de los datos	
Descripción:	Mediante el transporte de los datos se logra interceptar el cfd, o datos y se modifica o elimina la información ahi contenida. De igual manera se puede interceptar el CFD para provocar un ataque de DOS	
Precondiciones:	1.- Un atacante se encuentra escuchando el canal por el que se mandan los datos.2.- El atacante obtiene el CFD u otros datos, y modifica o elimina los datos que solo le convengan a el. Si modifica el CFD crea el sello digital con los datos nuevos generando una nueva factura válida.3. Otra opción es que el adversario robe por completo el CFD.3. Se reenvia los datos modificados al receptor	
Suposiciones:		
Post-Condiciones:	Peor Caso	Se modifican/eliminan los datos con exito.
	Medida de prevención deseada	-Transporte de los datos mediante un canal seguro
	Medida de detección deseada	-Cruce de facturas con el emisor
	Medidia de recuperación deseada.	
Perfiles de potenciales anti-actores:	Usuarios con conocimiento técnico agudo.	
Clientes y amenazas:	Clientes: Modificación de facturas sin consentimiento del contribuyente.	
Casos de uso relacionados:		
Amenazas relacionadas:	Alteración de la información, Revelación de la información, Suplantación de identidad,Denegación de servicio	
Recomendaciones arquitecturales:	-Contar con un canal seguro	

Cuadro D.7: Caso del Mal Uso MC-06

Numero:	MC-07	
Nombre:	Saltar el control de acceso de la aplicación.	
Alcance:	Generar una factura,Modificar información sensible del sistema como el login,crear clientes falsos.	
Prioridad:	Alta.	
Ambiente:	Aplicación	
Anti-Actores:	Usuarios no autorizados	
Niveles de derecho de acceso:	Admininstrador o usuario	
Puntos de entrada:	Aplicación.	
Atributos de seguridad afectados:	Autenticación	
Descripción:	El atacante realiza alguna de las siguientes acciones: ataque de diccionario,explota control de acceso débil, explota vulnerabilidad de S0/aplicación, para tratar de obtener acceso al dispositivo para posteriormente realizar una factura.	
Precondiciones:	Se cuenta con acceso al dispositivo Se accede a la aplicación por medio de control de acceso. Se realiza alguna de las siguientes acciones:taque de diccionario,explota control de acceso débil, explota vulnerabilidad de S0/aplicación. Se deben de tener folios precargados. 5. Se accede al modulo para generar facturas electrónicas, al de modificar login del sistema, o al de crear nuevo cliente.	
Post-Condiciones:	Peor Caso	Se crean cfd's, se modifica la información del cliente o se crea un cliente con éxito.
	Medida de prevención deseada	-Tener políticas de password seguras -Bloquear la aplicación -Digesto de la contraseña
	Medida de detección deseada	-Tener una bitacora de las acciones realizadas en la aplicación
	Medidia de recuperación deseada.	-Bloquear la aplicación despues de tres intentos
Perfiles de potenciales anti-actores:	Usuarios con conocimientos técnicos avanzados	
Clientes y amenazas:	Clientes: Creación de facturas sin conocimientos del contribuyente	
Casos de uso relacionados:		
Amenazas relacionadas:	Suplantación,Repudiación, Denegación de servicio,Elevación de privilegios.	
Recomendaciones arquitecturales:	Bitácora de la aplicación, digesto de la contraseña	
Recomendaciones arquitecturales:	Políticas de contraseñas seguras	

Numero:	MC-08	
Nombre:	Modificación de los comprobantes fiscales digitales.	
Alcance:	Compromente la integridad de los datos contenidos en la aplicación	
Prioridad:	Alta	
Ambiente:	Aplicación	
Anti-Actores:	Usuarios no autorizados	
Niveles de derecho de acceso:	Ninguno	
Puntos de entrada:	Hardware	
Atributos de seguridad afectados:	Autenticidad de la información	
Descripción:	Como se sabe los comprobantes digitales emitidos deben de ser guardados por 5 años, esta tarea es delegada al contribuyente por lo que en caso de que algún algoritmo criptográfico sea roto como se menciona en MC-05 podemos comprometer la seguridad de los CFD's almacenados.	
Precondiciones:	Se tiene acceso directo al contenedor del dispositivo. Se vulnera los CFD's ya que algun algoritmo de utilizado en la especificación del anexo 20 ha sido roto en un tiempo no mayor a 5 años.	
Post-Condiciones:	Peor Caso	Se vulnera los CFD's
	Medida de prevención deseada	-Tener previstos los cambios de algoritmos criptográficos.
	Medida de detección deseada	Ninguno
	Medidia de recuperación deseada.	Cambiar algoritmos criptográficos
Perfiles de potenciales anti-actores:	Usuarios con conocimientos técnicos avanzados	
Clientes y amenazas:	Clientes: Alteración de CFD almacenados a largo plazo	
Casos de uso relacionados:		
Amenazas relacionadas:	Modificación de la información	
Recomendaciones arquitecturales:	-Integrar un mecanismo de almacenamiento a largo plazo para evitar su modificación	

Cuadro D.8: Caso del Mal Uso MC-08

Numero:	MC-09	
Nombre:	Modificación/Eliminación de los datos respaldados en la base de datos local del dispositivo.	
Alcance:	Aplicación	
Prioridad:	Alta	
Ambiente:	Host	
Anti-Actores:	Usuarios no autorizados	
Niveles de derecho de acceso:	Ninguno	
Puntos de entrada:	Aplicación	
Atributos de seguridad afectados:	Confidencialidad,integridad	
Descripción:	Se accede por una aplicación diferente a la de facturación a la base de datos local donde se guarda la información y esta es modificada o eliminada.	
Precondiciones:	1.- Se cuenta con una aplicación que puede leer las base de datos locales del dispositivo.2.- Las base de datos para acceder a ella tienen un password debil o el que esta por defecto.	
Post-Condiciones:	Peor Caso	Se modifican/eliminan los datos con exito del reporte mensual
	Medida de prevención deseada	Contar con contraseñas fuertes para entrar a la base de datos
	Medida de detección deseada	
	Medidia de recuperación deseada.	
Perfiles de potenciales anti-actores:	Usuarios con conocimientos avanzados	
Clientes y amenazas:	Clientes: Eliminación de datos	
Amenazas relacionadas:	Modificación/Eliminación de la información	
Políticas de Recomendación:	Políticas de contrasenãs seguras en la BD.	

Cuadro D.9: Caso del Mal Uso MC-09

Numero:	MC-10	
Nombre:	Modificar/eliminar los datos del CFD, datos de clientes o usuarios ganando acceso directo hacia el dispositivo	
Alcance:	Compromete el funcionamiento de la aplicación.	
Prioridad:	Alta	
Ambiente:	Host	
Anti-Actores:	Integridad de la información,Disponibilidad	
Niveles de derecho de acceso:	Nivel usuario o administrador	
Puntos de entrada:	Host	
Atributos de seguridad afectados:	Autenticidad de los datos, integridad, disponibilidad.	
Descripción:	Mediante los siguientes ataques: robo del dispositivo, explotar un control de acceso débil o una vulnerabilidad en el sistema operativo se logra obtener acceso a parámetros susceptibles en la aplicación y estos son modificados.	
Precondiciones:	1.- El atacante obtiene el CFD u otros datos, y modifica o elimina los datos que solo le convengan a el. En el caso del CFD crea el sello digital con los datos nuevos generando una nueva factura válida.	
Post-Condiciones:	Peor Caso	Se modifican/eliminan los datos con exito.
	Medida de prevención deseada	-Los parametros sensibles deben estar cifrados(contenedor seguro)
	Medida de detección deseada	-Realizar función de autoprueba para comprobar integridad de datos.
	Medidia de recuperación deseada.	
Perfiles de potenciales anti-actores:	Usuarios con conocimiento técnico agudo.	
Clientes y amenazas:	Clientes: Modificación de facturas sin consentimiento del contribuyente.	
Amenazas relacionadas:	Alteración de la información, Revelación de la información, Suplantación de identidad.	
Recomendaciones arquitecturales:	-Contar con un contenedor seguro.-Tener una función de autoprueba.	

Cuadro D.10: Caso del Mal Uso MC-10

Numero:	MC-11	
Nombre:	Modificar/eliminar los datos del CFD datos clientes o usuarios ganando acceso directo hacia el servidor de respaldo	
Alcance:	Compromete el funcionamiento de la aplicación.	
Prioridad:	Alta	
Ambiente:	Servidor	
Anti-Actores:	Integridad de la información,Disponibilidad	
Niveles de derecho de acceso:	Nivel usuario o administrador	
Puntos de entrada:	Servidor	
Atributos de seguridad afectados:	Autenticidad de los datos, integridad, disponibilidad.	
Descripción:	Mediante los siguientes ataques: robo del servidor, mala administración de cuentas de usurario, o una contraseña débil en el servidor el usuario elimina o modifica los CFD o datos de clientes o usuarios	
Precondiciones:	1.- El atacante obtiene el CFD u otros datos, y modifica o elimina los datos que solo le convengan a el. En el caso del CFD crea el sello digital con los datos nuevos generando una nueva factura válida.	
Post-Condiciones:	Peor Caso	Se modifican/eliminan los datos con exito.
	Medida de prevención deseada	-Los parametros sensibles deben de estar cifrados (contenedor seguro)
	Medida de detección deseada	
	Medidia de recuperación deseada.	
Perfiles de potenciales anti-actores:	Usuarios con conocimiento técnico agudo.	
Clientes y amenazas:	Clientes: Modificación de facturas sin consentimiento del contribuyente.	
Amenazas relacionadas:	Alteración de la información, Revelación de la información, Suplantación de identidad.	
Recomendaciones arquitecturales:	-Contar con un contenedor seguro en el servidor	

Cuadro D.11: Caso del Mal Uso MC-11

Apéndice E

Arboles de ataque

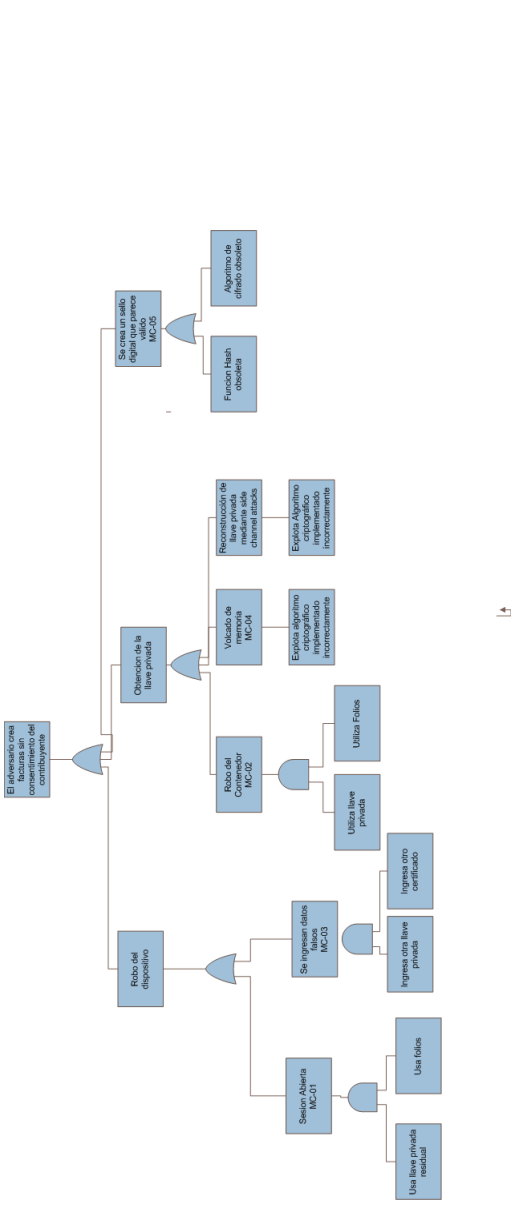


Figura E.1: Árbol de ataque: crear factura

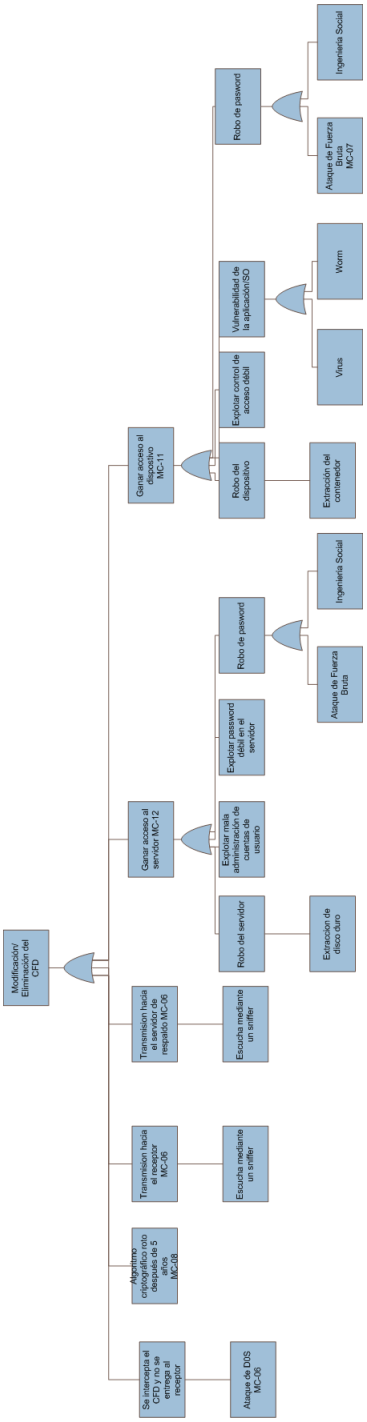


Figura E.2: Árbol de ataque: Modificar/Eliminar CFD

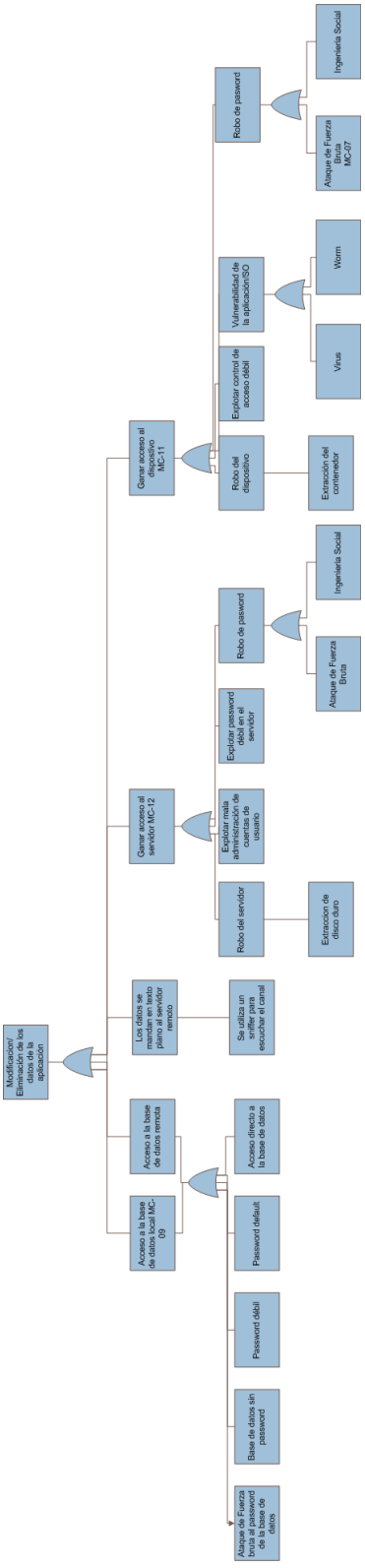


Figura E.3: Árbol de ataque: Modificar/Eliminar datos

Bibliografía

[1] SANS Password Policy, web page `www.sans.org`.

[2] Secretaría de Hacienda y Crédito Público. Diario Oficial de la Federación, Anexo 20 de la Resolución Miscelánea, 11 de junio 2010.

[3] Secure Coding Guidelines for the Java Programming Language Version 3.0, `http://www.oracle.com/technetwork/java/seccodeguide-139067.html`.

[4] Servicio de Administración Tributaria. SAT (México), Resolución Miscelánea Fiscal, 2010 .

[5] Servicio de Administración Tributaria. SAT (México), web page `http://www.sat.com.mx` .

[6] Simple XML Serialization, web page `http://simple.sourceforge.net/`.

[7] The Stride Threat Model, MSDN Library, web page. `http://msdn.microsoft.com/en-us/library/` .

[8] C. Adams and S. Farrell. Internet X. 509 public key infrastructure certificate management protocols, 1999.

[9] C. Adams and S. Lloyd. *Understanding PKI: concepts, standards, and deployment considerations*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2002.

[10] Deepa Padmanabhan Nancy R. Mead Ashwin Gayash, Venkatesh Viswanathan. SQUARE-Lite: Case Study on VADSoft Project. 2008.

[11] Mary Beth Chrissis, Mike Konrad, and Sandy Shrum. *CMMI Guidelines for Process Integration and Product Improvement*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[12] M. Graff and K. van Wyk. *Secure Coding: Principles and Practices. Sebastopol*. O'Reilly, 2003.

[13] M. Howard and D.E. Leblanc. *Writing secure code*. Microsoft Press Redmond, WA, USA, 2002.

[14] Michael Howard and Steve Lipner. *The Security Development Lifecycle*. Microsoft Press Redmond, WA, USA, 2005.

[15] Julia H. Allen and Dr. Robert and J. Ellison and Dr. Nancy and R. Mead and Cigital Inc. A look at software security engineering: A guide for project managers, 2008.

[16] J. Jurjens. Towards development of secure systems using UMLsec. *Fundamental Approaches to Software Engineering*, 2001.

[17] J. Jurjens. UMLsec: Extending UML for Secure Systems Development, UML 2002. *Lecture Notes in Computer Science*, 2460, 2002.

[18] J. Jurjens. Using UMLsec and goal trees for secure systems development. In *Proceedings of the 2002 ACM symposium on Applied computing*. ACM, 2002.

[19] B. Kaliski. Pkcs #5: Password-based cryptography specification version 2.0, 2000.

[20] N.R. Mead and T. Stehney. Security quality requirements engineering (SQUARE) methodology. In *Proceedings of the 2005 workshop on Software engineering for secure systems for building trustworthy applications*. ACM, 2005.

[21] Menezes, Alfred J. and Vanstone, Scott A. and Oorschot, Paul C. Van. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996.

[22] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X. 509 Internet public key infrastructure online certificate status protocol-OCSP, 1999.

[23] National Institute of standards and Technology. SECURITY REQUIREMENTS FOR CRYPTOGRAPHIC MODULES, FIPS 140-3, 2009.

[24] Christof Paar and Jan Pelzl. *Understanding Cryptography - A Textbook for Students and Practitioners.* Springer, 2010.

[25] G. Sindre and A.L. Opdahl. Eliciting security requirements with misuse cases. *Requirements Engineering*, 10(1), 2005.

[26] Ian Sommerville. *Software engineering (5th ed.)*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1995.

[27] W. Stallings. *Cryptography and network security*. Prentice Hall New Jersey, 2003.

[28] Douglas R. Stinson. *Cryptography: Theory and Practice, Third Edition (Discrete Mathematics and Its Applications)*. Chapman & Hall/CRC, November 2005.

[29] F. Swiderski and W. Snyder. *Threat modeling*. Microsoft Press Redmond, WA, USA, 2004.

[30] Dan Taylor and Gary McGraw. Adopting a software security improvement program. *IEEE Security and Privacy*, 3:88–91, 2005.

[31] W. Trappe, L. Washington, M. Anshel, and K.D. Boklan. Introduction to cryptography with coding theory. 2007.