Universidad de las Ciencias Informáticas

Facultad 1

Título: Firma Digital de Documentos utilizando Smart Cards.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autor: Roberto Quiñones Bondartchuk

Tutor: Ing. Alexei Zubizarreta Pérez

"Ciudad de La Habana. Junio, 2009"

Datos de contacto

Tutor: Ing. Alexei Zubizarreta Pérez.

Síntesis del Tutor: Informático, Instructor. Cinco años de experiencia en la implementación de proyectos de Gestión, dos años de experiencia en el área de Seguridad Digital.

Resumen

En el presente trabajo se hace un estudio de la de **firma digital** como mecanismo de seguridad y se desarrolla una aplicación multiplataforma para la **firma digital** de documentos en formato **PDF** utilizando **tarjetas inteligentes**, que garantiza la **autenticidad**, **integridad** y **no repudio** de la información almacenada en dichos documento. Además se hace uso del servicio de sellado de tiempo para garantizar la validez de la **firma digital** a largo plazo. Se describen las herramientas, tecnologías utilizadas y artefactos generados en el proceso de desarrollo. Igualmente se describe la arquitectura empleada. Para el desarrollo de la solución propuesta se tuvieron en cuenta los requerimientos del cliente, que finalmente se probaron mediante las pruebas de aceptación realizadas al término de cada iteración planificada.

Palabras claves: firma digital, PDF, tarjetas inteligentes, autenticidad, integridad, no repudio.

Índice

Índice de Tablas

Índice de Figuras

Introducción

Desde mediados del siglo pasado las sociedades han ido adentrándose en el inmenso mundo de la información. Volúmenes cada vez más crecientes son procesados, transmitidos y almacenados por el hombre. El papel, “soporte tradicional de datos”, cede su rol protagónico a nuevas formas de almacenamiento más duraderas, y con ellas surgen nuevos problemas de **seguridad.** ¿Cómo transmitir la información de manera segura? ¿Cómo saber que no fue alterada? y ¿Cómo saber si la fuente es confiable? son preguntas que se hacen a menudo.

La **criptografía** tiene las respuestas a esas preguntas. A través de algoritmos matemáticos y computacionales, esta es capaz de transformar una entrada de datos (secuencia de bits) legible en una salida de datos sin sentido directo para todos, excepto para la entidad deseada. Existen dos tipos de criptografía, simétrica y asimétrica, y ambas proporcionan **autenticidad**, **confidencialidad** e **integridad** a la información. Dentro de sus aplicaciones podemos contar con: cifrado de los canales de comunicación, certificados digitales y la **firma digital**.

El fraude, el robo de identidad y falta de confianza entre las partes de la comunicación, son problemas comunes en el mundo electrónico. La **firma digital**, como aplicación de la criptografía, provee los mecanismos para evitar dichos problemas. Al igual que la firma manuscrita al papel, la **firma digital**, provee a los **documentos electrónicos** las propiedades de **autenticación**, **no repudio** e **integridad**.

El uso de esta técnica ha crecido a nivel mundial. En sus inicios fue asumida por empresas norteamericanas como Dell, Intel, Hewlett-Packard y AT&T en sus proyectos pilotos(1) y hoy la **firma digital** es parte de la vida diaria de muchos habitantes del planeta. Gobiernos como el de Estados Unidos y los de los países de la Unión Europea fueron los primeros en aprobar el valor legal de la misma, luego otros países se fueron sumando a la lista, incluso muchos países de Latinoamérica. Empresas como Gemalto y Verisign proveen productos y soluciones de alto nivel para sectores públicos, empresas e Internet, basados en esta y otras tecnologías.

Muy ligado al empleo de la **firma digital**, están los **documentos electrónicos**, por ser estos los que guardan la información a proteger. Dichos documentos almacenan cualquier tipo de datos: texto, imagen, audio y video son los más usuales. Dentro de los formatos más conocidos se encuentra el **PDF** (Portable Document Format) el cual constituye un estándar abierto (ISO 32000-1:2008) y combina los tipos de datos antes mencionados. Además este formato es independiente de la plataforma o sistema operativo, puede ser generado utilizando varios

lenguajes de programación y ser creado con características especiales, como la accesibilidad para personas discapacitadas. El **PDF** es empleado en facturación y publicaciones electrónicas, emisión de contratos digitales y licencias en Internet.

Además la **firma digital** como técnica criptográfica puede ser implementada en dispositivos criptográficos; ejemplo de ello son las "smart cards" o **tarjetas inteligentes**. Estas tienen como antecedente las tarjetas de banda magnética, y tienen en particular que son tarjetas con un microchip incluido capaz de almacenar información de manera segura y de realizar operaciones criptográficas como el cifrado y la **firma digital**. El área de aplicación de estos dispositivos abarca desde la telefonía móvil e Internet hasta transacciones bancarias y documentos de identificación electrónicos. El empleo de la tecnología de **tarjetas inteligentes** para la **firma digital** de **documentos electrónicos** incrementa el grado de seguridad de la **firma digital**, por realizarse precisamente en un dispositivo seguro no se comprometen datos importantes para el proceso.

En Cuba, según(2), los principales esfuerzos en esta área de la tecnología han estado dirigidos al comercio electrónico. En 1999 se creó la Comisión Cubana de Comercio Electrónico presidida por los Ministerios del Comercio Exterior y el de la Sideromecánica y Electrónica. De forma paralela se establece al Ministerio del Interior como el organismo rector de los sistemas de protección criptográfica y prestación de estos servicios.

La Universidad de las Ciencias Informáticas (UCI) ha estado llevando a cabo el desarrollo de proyectos de identificación y seguridad con el uso de la tecnología de **tarjetas inteligentes** en pasaportes electrónicos. Además en el marco de los Acuerdos del ALBA, la universidad desarrolló una solución informática para las oficinas de Registros y Notarías de la República Bolivariana de Venezuela, dicha solución contó con un componente para generación y verificación de **firma digital** de documentos.

Soluciones para la administración pública, comercio y gobierno electrónico están dentro de las líneas investigativas y posibles líneas de producción de proyectos en la UCI; dichas soluciones son impensables sin la gestión eficiente de la seguridad. Es por ello que se hace necesario el estudio y desarrollo de productos informáticos que incorporen elementos como la **firma digital** y el uso de dispositivos seguros como las **tarjetas inteligentes**.

Con el análisis de lo antes expuesto y con el fin de darle solución a la problemática existente, este trabajo plantea el siguiente problema científico: *¿Cómo garantizar la **autenticidad**, **integridad** y **no repudio** de **documentos electrónicos** en formato **PDF**?*

Para dar solución al problema científico se asume como objeto de estudio: *la **firma digital** de **documentos electrónicos*** y como campo de acción: *la **firma digital** avanzada de **documentos electrónicos** en formato **PDF** utilizando **tarjetas inteligentes**.*

Por lo que el objetivo general de esta investigación es: *Desarrollar una aplicación multiplataforma para la **firma digital** de documentos en formato **PDF**, utilizando **tarjetas inteligentes**, para garantizar la **autenticidad**, **integridad** y **no repudio** de información almacenada en dichos documentos.*

A partir del análisis del objetivo general se derivaron los siguientes objetivos específicos:

- Identificar y valorar diferentes aplicaciones de **firma digital**.
- Analizar los estándares para el trabajo con **tarjetas inteligentes** y firmas digitales.
- Diseñar la aplicación para la **firma digital** de documentos en formato **PDF** utilizando **tarjetas inteligentes**.
- Implementar la aplicación para la **firma digital** de documentos en formato **PDF** utilizando **tarjetas inteligentes**.
- Evaluar la calidad y cumplimiento con estándares de la aplicación para la **firma digital** de documentos en formato **PDF** utilizando **tarjetas inteligentes**.

Para darle cumplimiento a los objetivos antes mencionados se definen las siguientes tareas de investigación:

1. Revisión de aplicaciones o soluciones similares de **firma digital** de documentos, propietarias, libres y desarrolladas en diferentes tecnologías.
2. Estudio de estándares de **firma digital** y dispositivos seguros.
3. Selección de algoritmos criptográficos de **firma digital**.
4. Selección de estándares para la interacción con **tarjetas inteligentes**.
5. Selección de tecnologías y herramientas a usar para el desarrollo de la aplicación.
6. Definición de los requerimientos de la aplicación para la **firma digital** de documentos en formato **PDF** utilizando **tarjetas inteligentes**.
7. Desarrollo del análisis de la aplicación para la **firma digital** de documentos en formato **PDF** utilizando **tarjetas inteligentes.**
8. Implementación de la aplicación para la **firma digital** de documentos en formato **PDF** utilizando **tarjetas inteligentes.**

9. Ejecución de pruebas de caja negra y caja blanca a la aplicación para la **firma digital** de documentos en formato **PDF** utilizando **tarjetas inteligentes.**

Esta investigación defiende la idea de que: Desarrollando una aplicación para la **firma digital** de documentos en formato **PDF** usando **tarjetas inteligentes** se garantiza la **integridad**, **no repudio** y **autenticidad** de la información almacenada en dichos documentos.

Para el desarrollo exitoso de la investigación se propone la utilización de métodos investigativos teóricos:

- **Histórico lógico:** Permite estudiar de forma analítica la trayectoria histórica real de los fenómenos, su evolución y desarrollo.
    - Usado con el objetivo de estudiar la evolución de la **firma digital**, dispositivos criptográficos, estándares de criptografía y **documentos electrónicos**.
- **Analítico-sintético:** Facilita el entendimiento del fenómeno en el que se trabaja, es más útil la división de este en diferentes fases, y de esta forma descubrir sus características generales, lo que ayuda a seguir una correcta investigación.
    - A partir de la información obtenida se hace necesario organizarla y sintetizarla para elaborar una estructura adecuada.
- **Inductivo-Deductivo:** Se hace uso de deducciones para llegar a tener una visión clara de lo que se quiere hacer y adquirir así nuevos conocimientos.
    - Partiendo de la idea a defender planteada se busca información asociada con el tema para proponer una solución a la problemática existente.

y empíricos:

- **Entrevista:** Es una conversación planificada entre el investigador y el entrevistado para obtener información.
    - Tiene como objetivo obtener la mayor información posible sobre los requisitos que el cliente desea.

Con esta investigación se pretende obtener los siguientes resultados:

- Aplicación para la **firma digital** de documentos en formato **PDF** que pueda ser comercializada.

Estructura de la tesis

El presente trabajo está estructurado en tres capítulos los cuales se describen a continuación:

Capítulo 1: Fundamentación Teórica

Se presentan los conceptos fundamentales relacionados con la firma digital, los documentos electrónicos en formato PDF y el uso de dispositivos criptográficos vinculados al proceso de firma digital. Al finalizar el capítulo se realiza un estudio de aplicaciones de software para la firma digital.

Capítulo 2: Propuesta de Solución

Se presentan las fases de exploración y planificación definidas por la metodología XP para dar solución al problema científico. Se seleccionan las herramientas y tecnologías para desarrollar la solución. Se identifican las diferentes historias de usuarios y los requerimientos no funcionales, se realiza el plan de iteraciones y plan de entregas.

Capítulo 3: Implementación de la Solución

Se da cumplimiento a los planes trazados a través de las fases de iteraciones a primer liberación y producción, se codifica la solución diseñada y finalmente se realizan las pruebas de aceptación con el cliente.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

El objetivo principal que se persigue con la elaboración de este capítulo es la comprensión de las bases teóricas de la firma digital como mecanismo para garantizar la autenticidad de documentos electrónicos. Además se hace un estudio del estado del arte de los algoritmos fundamentales que dan soporte a la firma digital, así como los dispositivos criptográficos más usados en este proceso. Finalmente se analizan otras aplicaciones para la firma digital de documentos en formato PDF y se presenta la propuesta de la solución al problema científico rector de esta investigación.

1.2 Criptografía

La criptografía se define como "arte de escribir con clave secreta o de manera enigmática". Esta data de la época del César y desde entonces ha sido su objetivo preservar el secreto de la información en la comunicación entre dos o más entidades.

Importantes cambios a través de la historia han convertido estas técnicas de escritura en verdaderos sistemas de seguridad; ejemplo de ello fue la máquina Enigma[1], utilizada por los alemanes en La Segunda Guerra Mundial. Hasta ese momento, los algoritmos utilizados en los criptosistemas eran los que conocemos hoy como "algoritmos de llave privada" o "algoritmos simétricos", basados en sustituciones mono alfabéticas, poli alfabéticas y transposiciones. Todo el secreto estaba en conocer la clave utilizada para cifrar el mensaje. No fue hasta los años 70 del siglo XX que surge la criptografía moderna con la aparición de los "algoritmos de llave pública" o "algoritmos asimétricos", creados por Whitfield Diffie[2] y Martin E. Hellman[3]; esta nueva teoría ligada al desarrollo vertiginoso de la computación, hizo que surgieran novedosas aplicaciones de la criptografía como el cifrado de los canales de comunicación, la autenticación, los certificados digitales y la firma digital entre otros.

Una manera fácil de entender qué es la criptografía, es a través de un ejemplo práctico: Supongamos que Ángel desea mandar una mensaje a Beatriz sin que Carlos sepa que dice el

[1] Máquina creada en 1923 por un ingeniero alemán llamado Arthur Scherbius y diseñada para facilitar las comunicaciones seguras. Esta utilizaba un sistema de rotores para codificar los mensajes tecleados en la máquina.
[2] Whitfield Diffie, en 1965 se graduó de matemático en el Instituto Tecnológico de Massachusetts. En 1976 publicó junto a Martin Hellman "New Directions in Cryptography", convirtiéndose en pionero de la criptografía asimétrica.
[3] Martin E. Hellman, se graduó en la Universidad de Nueva York en 1966 con el título de Ingeniería Eléctrica. Hellman es famoso por ser el inventor junto a Diffie de la criptografía de llave pública.

mensaje; Ángel parte del texto plano y una información que solo él y Beatriz conocen (clave secreta), luego aplica un proceso al mensaje para transformarlo en información ilegible (cifrado) y lo envía a Beatriz, ella para obtener el mensaje original realiza un proceso inverso (descifrado) utilizando la clave secreta; si Carlos interceptara el mensaje no sería capaz de saber qué dice, por el simple hecho de desconocer qué proceso de transformación se le ha aplicado al mismo. Ahora bien, si Carlos logra descubrir el proceso que se aplicó, aun así le faltaría saber la clave secreta, sin la cual sería difícil descubrir el mensaje (ver Figura 1).

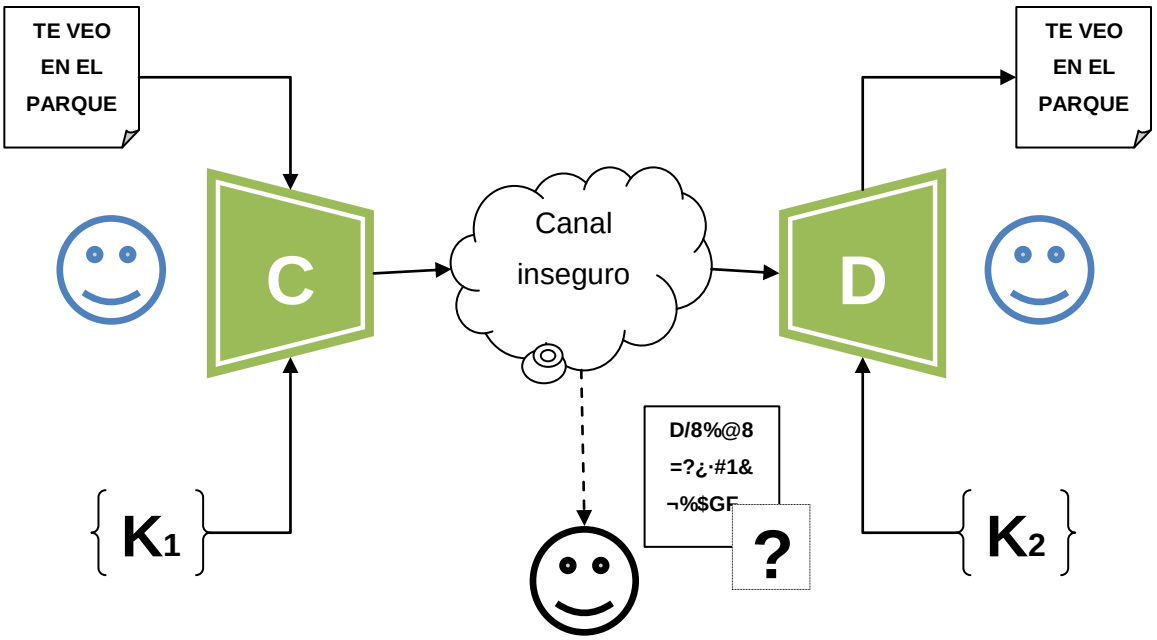

Figura 1 Sistema Criptográfico

K = clave secreta, C = cifrado, D = descifrado.

De manera general, el proceso completo de cifrado y descifrado en una comunicación entre dos entidades se puede representar con las formulas siguientes:

$$C\ (M, K_1) = M_c$$

$$D\ (M_c, K_2) = M$$

Donde:

C: algoritmo de cifrado, recibe como entrada **M**: mensaje original sin cifrar y $\mathbf{K_1}$: clave secreta (a partir de este momento llave privada), dando como resultado $\mathbf{M_c}$: mensaje cifrado.

D: algoritmo de descifrado, recibe como entrada $\mathbf{M_c}$: mensaje cifrado y $\mathbf{K_2}$: clave secreta (a partir de este momento llave privada), dando como resultado **M**: mensaje original sin cifrar.

1.2.1 Criptografía simétrica

La criptografía simétrica también conocida como criptografía de llave privada, está basada en la criptografía clásica y en los principios de confusión y difusión de Claude Shannon[4]. Lo distintivo de la criptografía simétrica es que los algoritmos de cifrado y descifrados son inversos y utilizan la misma llave; apoyándonos en la fig. 1, K1 = K2 = K por lo tanto:

$$\mathbf{D\ (C\ (M,\ K),\ K) = M}$$

En este tipo de criptografía se usan dos clases de algoritmos: algoritmos de cifrado por bloques y algoritmos de cifrado por flujo, a continuación se explicará cada tipo.

1.2.1.1 Cifrado por bloques

En principio el método consiste en dividir el mensaje en bloques de tamaño fijo, y aplicar la función de cifrado a cada uno de ellos(3). Generalmente el tamaño de los bloques es de 8bytes o 16bytes(4). Una práctica bastante común es realizar un XOR entre el bloque a cifrar y el último bloque cifrado, luego se cifra el resultado de esta operación.

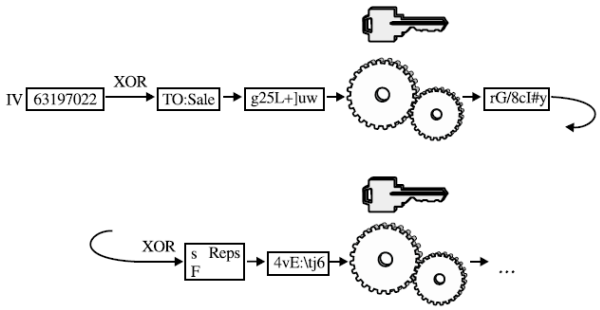

Figura 2 Cifrado por bloque encadenados.

[4] Claude Elwood Shannon (1916 - 2001), ingeniero eléctrico y matemático, recordado como "el padre de la teoría de la información".

1.2.1.2 Cifrado por flujo

Los algoritmos implementados de este modo, constan de un generador pseudo-aleatorio capaz de generar, a partir de una semilla[5] (llave privada), secuencias de bytes aleatorias, las cuales se utilizan para cifrar mensajes simplemente aplicando la función XOR entre el texto en claro y la secuencia de bytes generada. Todo aquel que conozca, la llave privada podría reconstruir la secuencia pseudo-aleatoria y de esta forma descifrar el mensaje.

Según (4), el cifrado por flujo es generalmente más rápido que el cifrado por bloques, y mucho más fácil de implementar; pero a diferencia del cifrado por bloques, las llaves privadas no son reutilizables y está dado por el hecho de que se utilizan secuencias aleatorias para cifrar los mensajes.

1.2.1.3 Algoritmos de cifrado simétrico

DES *Digital Encryption Standard*: fue creado por la IBM a principios de los años 70 y en 1976 fue adoptado como estándar por el Gobierno de los Estados Unidos para el cifrado de comunicaciones no clasificadas(3). Es un algoritmo de cifrado por bloques y la llave es de 64bit de los cuales solo 56bit son útiles. Con el desarrollo de la computación se demostró que era viable un ataque por fuerza bruta contra este algoritmo, y en una conferencia de la RSA Data Security[6] en 1999, la llave de un mensaje cifrado con DES fue rota en menos de 24 horas(4).

TripleDES: variante de DES para mejorar su fortaleza. Se aplica DES tres veces por lo que la longitud de la llave se triplica, o lo que es lo mismo, usar tres llaves de 56bit. Este algoritmo es más fuerte que DES pero tres veces más lento.

AES *Advanced Encryption Standard*: se da a conocer en el 2000 como resultado de un concurso lanzado por el Instituto Nacional de Estándares y Tecnología (NIST), en busca de un sustituto para DES y TripleDES. Este algoritmo utiliza cifrado por bloques y maneja llaves de longitudes de 128, 192 y 256bit(5) y está considerado uno de los algoritmos más seguros de la actualidad.

RC4: según (3) creado en 1987 por RSA Data Security. Es un algoritmo propietario y su código no ha sido publicado de manera oficial, aunque en 1994 alguien lo publicó anónimamente en

[5] Se le llama semilla a los parámetros iniciales de los mecanismos para generar números aleatorios.
[6] Actualmente RSA Security. Empresa líder en soluciones de seguridad informática.

internet. Usa cifrado por flujo y permite llaves de tamaño variable. Ahora es parte del protocolo SSL[7].

1.2.2 Criptografía asimétrica

En 1976, Whitfield Diffie y Martin E. Hellman describieron la idea de desarrollar un algoritmo de cifrado que utilizara dos llaves; una llave sería pública y la otra secreta o privada. Esto permitiría a las personas cifrar un mensaje usando la llave pública, y solo el dueño de la llave privada podría descifrarlo. Este novedoso cambio eliminaba el problema del intercambio de llaves, presente en la criptografía simétrica, además potenció la aparición de un sinnúmero de aplicaciones(6).

Volviendo al ejemplo práctico: ahora si Ángel desea mandar un mensaje a Beatriz, cifra el mensaje con la llave pública de Beatriz, y ella sería la única capaz de descifrarlo utilizando su llave privada (ver Fig. 3), por lo tanto $\mathbf{K_1 \neq K_2}$.

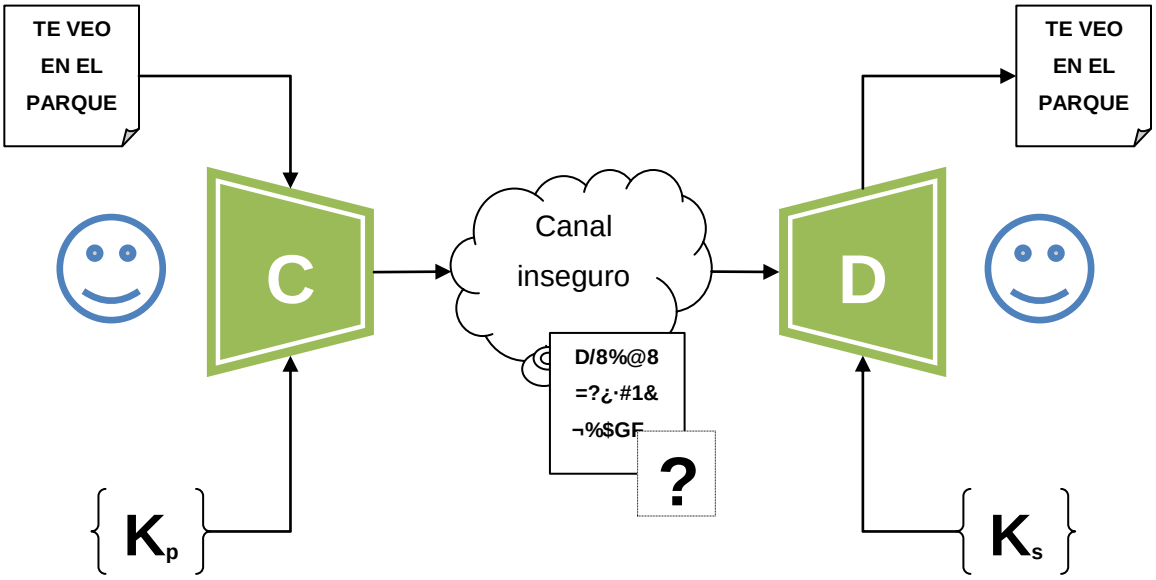

Figura 3 Sistema criptográfico asimétrico
K_p = llave pública de Beatriz K_s = llave secreta o privada de Beatriz C = cifrado D = descifrado

Los algoritmos de llave pública tienen un fuerte basamento matemático y consisten en problemas de difícil solución como la factorización de números primos, logaritmos discretos y la teoría de curvas elípticas. A continuación se explicarán algunos detalles de los algoritmos más usados.

[7] SSL (Secure Sockets Layer) es un protocolo criptográfico que proporciona comunicaciones seguras a través de una red, comúnmente Internet.

1.2.2.1 Algoritmos de cifrado asimétrico

RSA: Este algoritmo fue propuesto por Ron **R**ivest[8], Adi **S**hamir[9], y Len **A**dleman[10] en 1978, es el más conocido y versátil de los algoritmos de llave pública usados actualmente(6). Es apropiado para las operaciones de cifrado y firma digital .Su seguridad está basada en la dificultad de factorizar números enteros muy grandes y por el grado de avance en esta área de las matemáticas, se sugiere el empleo de llaves de al menos 1024bit. El proceso de cifrado y descifrado se puede representar matemáticamente de la siguiente forma:

Cifrado: $C = M^e \ mod \ n$

Descifrado: $M = \ C^d \ mod \ n$

Donde C: texto cifrado, M: texto claro, e: llave pública, d: llave privada,

n: Módulo público $= p * q$

$p, \ q$: Números primos secretos

DH *Diffie-Hellman*: Es un algoritmo asimétrico que se emplea fundamentalmente para acordar una llave común entre dos entidades (llave de sesión), a través de un canal de comunicación inseguro. La ventaja de este sistema es que no son necesarias llaves públicas en el sentido estricto, sino una información compartida por los dos comunicantes(3). Este procedimiento es conocido como "acuerdo de llaves". La seguridad del algoritmo está dada por la dificultad de computar logaritmos discretos(7), problema muy parecido al planteado por **RSA**. Se aconseja que las llaves sean de una longitud no menor que 1024bit.

ECDH *Elliptic Curve Diffie-Hellman*: Es igual que **DH**, pero sustentado en la teoría de las curvas elípticas sobre un conjunto discreto, también de difícil solución. Su finalidad, al igual que **DH**, no es el cifrado sino el intercambio de llaves de sesión. Se aconsejan llaves de 192 a 256bit.

Los algoritmos antes descritos resuelven el problema del intercambio de llaves, pero ¿cómo saber cuál es el mejor? El uso de **RSA** y **DH** están bastantes difundidos y son parte de varios

[8] Ronald L. Rivest obtuvo la licenciatura en matemáticas en la universidad de Yale en 1969. Junto con Adi Shamir y Len Adleman, crea el algoritmo RSA. Además es creador de otros algoritmos como RC2, RC4, RC5 y las funciones hash MD2, MD5, SHA y SHA-1.

[9] Adi Shamir es un criptógrafo Israelí. Fue uno de los inventores del algoritmo RSA (junto con Rivest y Adleman), y ha hecho numerosas contribuciones a los campos de la criptografía y las ciencias de la computación.

[10] Leonard Adleman es profesor en ciencias de la computación y biología molecular de la Universidad del Sur de California. Conocido por ser uno de los inventores de algoritmo RSA en 1977 y de la computación por ADN.

protocolos de seguridad como PGP[11], OpenPGP[12], S/MIME[13] y SSL[14] sobre todo para el establecimiento y empaquetado de llaves de sesión, cifrado y autenticación con firma digital en caso del **RSA**, sin embargo **ECDH** no goza este privilegio debido a que solo unos pocos científicos se han dedicado al estudio de la criptografía de curvas elípticas y las aplicaciones que lo implementan generalmente lo utilizan internamente, es decir no se comunican con otras aplicaciones. En cuanto a seguridad, muchos especialistas se inclinan por **RSA** ya que **DH** y **ECDH** son susceptibles a ataques de tipo *hombre en el medio*[15]. **RSA** es el algoritmo recomendado al no poseer problemas de seguridad y ser más versátil que **DH** y **ECDH**.

1.2.2.2 Funciones Hash

Las funciones hash son usadas generalmente para probar que la información trasmitida no ha sido alterada. Una función hash *H* recibe como entrada un mensaje *m* y da como salida el valor hash *h* de *m*, *h = H(m)*; m de longitud variable y h de longitud fija. Esto significa que una secuencia de bytes de cualquier tamaño, después de aplicarle una función de este tipo, dará como resultado una secuencia de una longitud fija de bytes. Estas funciones también son conocidas como funciones resumen o digesto y tienen la característica de que teniendo un resumen *h* resulta muy difícil hallar el mensaje *m* que lo produjo.

[11] PGP (Pretty Good Privacy) es un criptosistemas que combina criptografía simétrica y asimétrica y es usado en el cifrado de mensajes de correo electrónico y datos en general.
[12] OpenPGP es un protocolo para el cifrado de mensajes de correo electrónico basado en PGP.
[13] S/MIME (Secure / Multipurpose Internet Mail Extensions) es un estándar para criptografía de clave pública y firmado de correo electrónico.
[14] SSL (Secure Sockets Layer) es un protocolo de comunicación segura a través de una red de computadoras. Utiliza algoritmos criptográficos para garantizar la integridad i confidencialidad de la comunicación.
[15] Hombre en el medio, es un tipo de ataque específico a algoritmos de intercambio de llaves, donde precisamente el intercambio de llaves ocurre a través del atacante que suplanta las identidades de los involucrados en el intercambio.

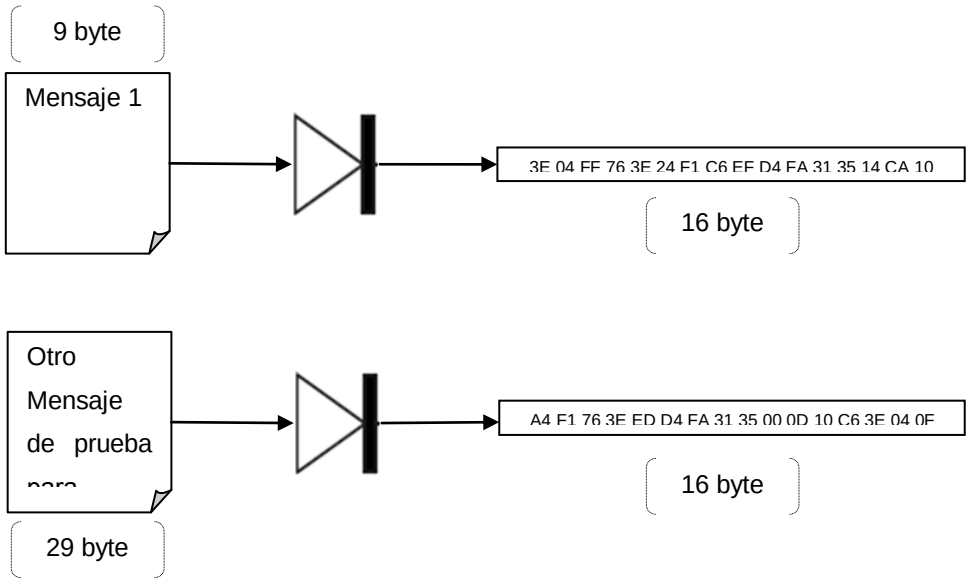

Figura 4 Función Hash.

¿Qué pasa cuando dos mensajes diferentes producen el mismo resumen? En este caso se estará en presencia de una colisión y es tan improbable como encontrar dos personas con las mismas huellas dactilares. Una función resumen se considera buena, cuando no genera colisiones y es irreversible.

Las funciones hash más importantes según (4) son:

- **MD2**: Creado por Ron Rivest como versión mejorada de otro algoritmo suyo, **MD**. **MD2** produce resúmenes de 128bit (16byte). MD2 fue ampliamente usado pero con el transcurso de los años se encontraron algunos defectos. Varias clases de mensajes producían colisiones. Hoy en día este algoritmo no se usa.

- **MD5**: También creado por Rivest el **MD5** es más rápido y fuerte que **MD2** y se convirtió en el algoritmo más usado de este tipo. También produce resúmenes de 128bit (16byte). Los estudios de este algoritmo han detectado posibles fayas de seguridad dadas por su estructura interna, aunque no se haya dado el caso de ninguna colisión.

- **SHA-1**: Este algoritmo es bastante parecido al **MD5**, pero a diferencia de este último, Rivest puso su empeño en el diseño de su estructura interna. **SHA-1** es más fuerte internamente que **MD5** y produce resúmenes más largos, 160 bits (20bytes),

característica que también le da fortaleza. **SHA-1** ha sobrevivido el criptoanálisis y actualmente es la función hash recomendada por la comunidad de criptografía. También existen variantes de este algoritmo que generan resúmenes de 256bit (32byte), 384bit (48byte) y 512bit (64byte).

1.3 Firma digital

La firma digital es una de las aplicaciones de la criptografía asimétrica de mayor éxito y es usada para establecer la autenticidad e integridad de mensajes y documentos electrónicos. La verificación de la firma digital permite determinar cuando un mensaje o documento ha sido alterado. Además tiene la propiedad de que solo puede ser producida de manera correcta por una entidad, y ser verificada por cualquiera que reciba el mensaje o documento firmado digitalmente(6).

1.3.1 Proceso de firmado

El proceso de la firma digital utiliza las ya mencionadas funciones hash como **SHA-1** y **MD5** y algoritmos de cifrado de llave pública como **RSA**. La figura a continuación muestra su funcionamiento.

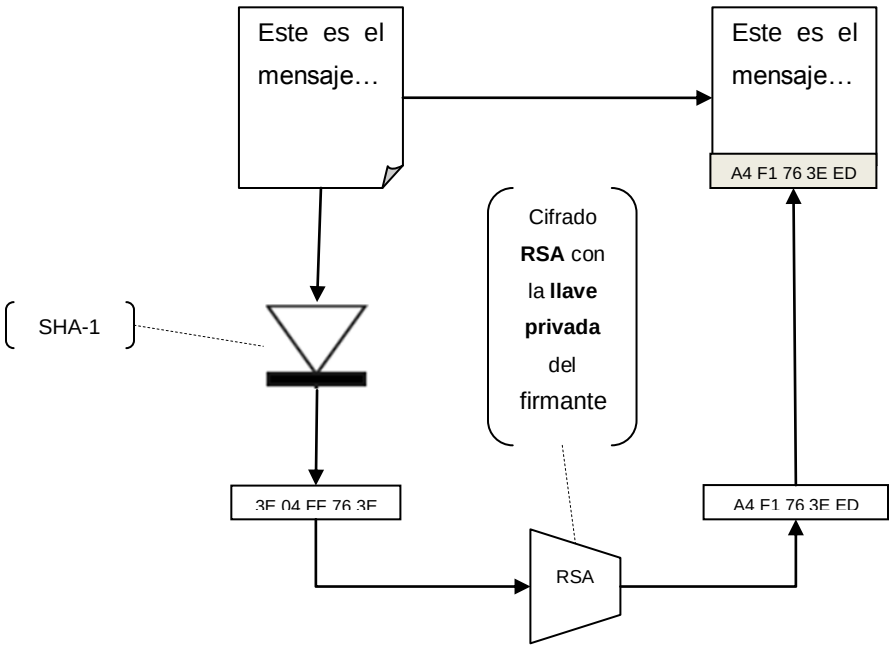

Figura 5 Proceso de Firma Digital.

Primero, la entidad que generará la firma, aplica una función hash al mensaje o documento electrónico que desea firmar; el resultado de la operación es cifrado con la llave privada de la entidad firmante, por último se agrega el resultado del cifrado al mensaje.

1.3.2 Proceso de verificación

El proceso de verificación de la firma digital es bastante sencillo; primero se separa la firma del documento y se le aplica el proceso de descifrado con la llave pública del firmante, el resultado de esta operación es el resumen o hash del documento; luego aplicamos la función hash al documento sin la firma y comparamos los dos hash obtenidos, si los dos hash son iguales, el documento no ha sido modificado, en otro caso no se estará en presencia de un documento auténtico (Ver Figura 6).

Las fuentes consultadas coinciden en que la firma digital provee a los documentos electrónicos de las siguientes características:

- **Autenticación**: el receptor del mensaje es capaz de verificar el origen del mismo.
- **No repudio**: el emisor del mensaje no puede negar que lo ha firmado y enviado.
- **Integridad**: el receptor del mensaje es capaz de verificar que el mensaje no haya sido alterado.

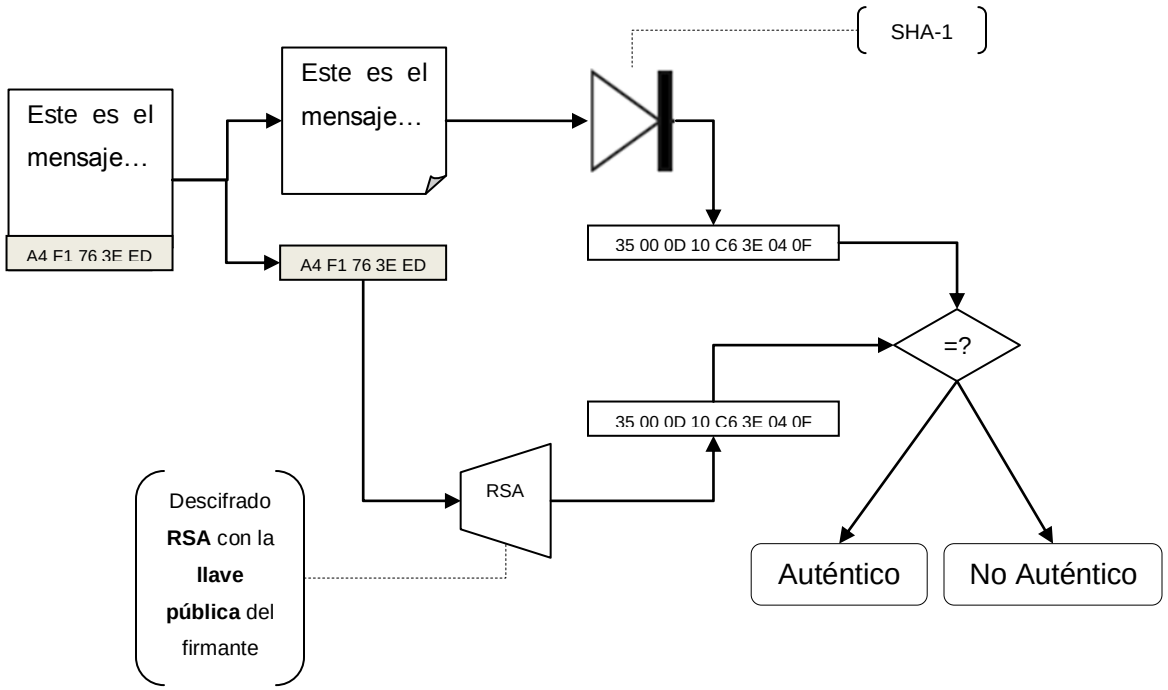

Figura 6 Proceso de verificación de la Firma Digital.

1.3.3 Algoritmos de Firma Digital

Actualmente muchas aplicaciones de seguridad que implementan la firma digital soportan varios de los mecanismos existentes, aunque no todos sean utilizados con igual frecuencia. Entre los más usados contamos con **SHA-1/RSA** y **DSA** (*Digital Signature Algorithm*); otros como **ECDSA** (*Elliptic Curve DSA*) son incorporados gradualmente debido a lo novedoso que resulta el campo de la criptografía de curvas elípticas. A continuación algunas de las características, según (4), de estos mecanismos:

- **SHA-1/RSA**: Este método combina **SHA-1** como función hash y **RSA** como algoritmo de cifrado, y el proceso de firma y verificación es de manera semejante al descrito en este epígrafe. Además el proceso de verificación resulta más rápido que el proceso de firmado, característica que lo hace superior a otros métodos como **DSA**, debido que lo más común es que un mensaje o documento electrónico sea firmado una vez y verificado varias veces. Actualmente es el método de firma digital más utilizado por las aplicaciones criptográfica y es soportado por muchos protocolos de seguridad.

- **DSA** *Digital Signature Algorithm*: Este algoritmo fue publicado en 1994 formando parte del estándar **DSS** (*Digital Signature Standard*)[16] adoptado por el Gobierno de Estados Unidos como estándar para el proceso de firma digital. Está basado en el problema de los logaritmos discretos y en el criptosistema ElGamal[17] y a diferencia del algoritmo **RSA** que se utiliza en las operaciones de cifrado y firma digital, **DSA** solo puede realizar la firma digital. Dentro de las características de este algoritmo tenemos la rapidez del proceso de firma respecto al proceso de verificación, el uso de **SHA-1** como función hash y las llaves de 1024bit.

- **ECDSA** *Elliptic Curve DSA*: **ECDSA** es una versión, basada en la teoría de curvas elípticas, del algoritmo **DSA**. Es más rápido que **RSA** y **DSA** pero actualmente muy pocas aplicaciones criptográficas lo incorporan entre sus algoritmos. Al igual que **DSA** forma parte del estándar **DSS** (Digital Signature Standard). Por ser un algoritmo de curva elíptica, la longitud de las llaves son mucho más cortas que en **RSA** y **DSA**, y es aconsejable utilizar al menos 192bit.

[16] DSS (Digital Signature Standard) fue adoptado por el gobierno de los Estados Unidos en 1994 como estándar para la firma digital. La especificación incluye los algoritmos DSA, SHA-1, RSA y ECDSA.
[17] ElGamal es un criptosistemas de llave pública basado en el problema de los logaritmos discretos y puede ser usado para el cifrado y la firma digita.

1.4 Certificados Digitales

En el proceso de firma y verificación existe un problema que no podemos dejar de mencionar. Cualquiera que desee verificar la validez de un mensaje o documento firmado, necesita de una llave pública apropiada. La llave pública no puede ser enviada sin algún tipo de protección, y de ser así el receptor del mensaje no podría verificar la validez de la llave. La manera de resolver este problema es que la lleve pública sea firmada por un tercero confiable, este tercero confiable es lo que llamamos autoridad certificadora (CA). La combinación de la llave pública firmada por una CA, la firma digital y otro grupo de datos, es lo que se conoce como certificado digital(6).

Una CA se encarga además de manejar peticiones de certificados, y lleva el control de los certificados existentes. Un certificado está ligado unívocamente a una persona u organización, es como un documento de identificación. Cuando una CA emite un certificado, lo hace en un medio seguro que contiene el par de llaves: la llave privada con la cual el propietario de certificado podrá realizar operaciones de cifrado y firma digital, y la llave pública firmada digitalmente por la CA garantizando la autenticidad de la llave.

Cuando la llave privada correspondiente a un certificado por alguna razón queda comprometida, el propietario del mismo informa a la CA, la cual revoca el certificado; es decir lo hace invalido.

1.5 Sellado de tiempo

Las firmas digitales suministradas a través de la utilización de la tecnología de llave pública se pueden poner en tela de juicio por una simple razón: si el firmante de un documento particularmente importante (por ejemplo, un acuerdo de préstamo) más tarde desea repudiar su firma, puede deshonestamente informar el compromiso de su llave privada y solicitar que sea revocada. Un verificador después no será capaz de certificar si la firma sucedió antes o después de la revocación (4). La situación antes descrita puede ser evitada mediante el uso del sellado de tiempo provisto por una autoridad de sellado de tiempo (TSA).

La función de una TSA es el sellado de tiempo de un dato para establecer evidencia de la existencia de ese dato antes de una fecha/hora determinada. Esto puede ser usado, por ejemplo, para verificar si una firma digital fue aplicada a un documento antes de que el certificado correspondiente fuese revocado, dando la posibilidad de que certificados revocados sean usados para verificar firmas creadas previamente a la fecha de revocación de los mismos(8).

La importancia del sellado de tiempo se pone de manifiesto cuando existe necesidad del uso legal de documentos electrónicos en un largo período de tiempo. Este mecanismo permite verificar la validez de un documento firmado después que el certificado correspondiente haya sido revocado. Además garantiza el no repudio de la firma digital (4).

1.6 Dispositivos Seguros

La seguridad de soluciones para la firma digital de mensajes y documentos electrónicos, pueden estar basadas en software, hardware o una combinación de ambos, en dependencia del nivel de seguridad requerido por la solución. Generalmente las soluciones basadas en hardware son más seguras, debido a que el manejo de las llaves de firma está encapsulado por la lógica del hardware.

Existen varios dispositivos criptográficos (soluciones basadas en hardware) capaces de realizar, entre otras operaciones criptográficas, la firma digital de mensajes o documentos electrónicos de manera segura. Este tipo de dispositivos incorpora componentes electrónicos especializados para garantizar la seguridad física del equipo y al mismo tiempo la lógica, creando un ambiente seguro de almacenamiento de las llaves mediante la utilización de funciones criptográficas. Entre estos dispositivos están los **HSM** y las **tarjetas inteligentes**.

HSM (Hardware Security Module), según (9), es dispositivo que genera, almacena y protege llaves criptográficas tanto simétricas como asimétricas y acelera el tiempo de proceso necesario para realizar operaciones criptográficas. Posee características tales como la posibilidad de detectar la manipulación del interior del dispositivo, un borrado completo de los mismos cuando se detecta la manipulación o el uso de sistemas operativos seguros. Generalmente están vinculados a servidores de firma digital, autoridades certificadoras (CA) y autoridades de sellado de tiempo (TSA).

Tarjetas Inteligentes, según (6), son tarjetas que surgieron en la década del 70 con el desarrollo de la microelectrónica. Constan de un microchip incrustado en un soporte de plástico de unos pocos centímetros cuadrados, que permite el almacenamiento y procesamiento de la información. Además poseen una capacidad de procesamiento que les permiten realizar operaciones criptográficas como el cifrado de la información y la **firma digital**. Las **tarjetas inteligentes** almacenan de manera segura la llave privada asociada a un certificado digital emitido por una CA.

Actualmente el uso de estas tarjetas abarca las áreas de salud, seguridad en Internet, sistemas de pago electrónico, sistemas de identificación y acceso, telefonía móvil, cifrado de datos y **firma digital** de documentos.

1.7 Documentos PDF

El surgimiento de los documentos electrónicos fue propiciado por la ineficiencia en el manejo de grandes volúmenes de información soportados en papel; además del gasto de recursos dedicados a la búsqueda, reproducción y recuperación de la información. Con el uso de los documentos electrónicos se facilitó el almacenamiento de la información y el aumento en la velocidad de transmisión de la misma. Sin embargo, el prescindir del papel creó el problema de la existencia de múltiples formatos incompatibles.

Estas incompatibilidades estaban dadas por el uso de diferentes sistemas operativos y software para crear documentos electrónicos del mismo tipo. Para resolver este problema, muchas compañías de software distribuyeron visualizadores gratuitos, que pueden leer los documentos electrónicos generados por sus programas de edición, que generalmente no son gratuitos, un buen ejemplo es Adobe Reader.

En el año 1993 Adobe Systems desarrolla el formato **PDF** (Portable Document Format), creado pensando en la oficina con información 100% digital(10). Este nuevo formato además de texto, incorporaba imágenes y gráficos vectoriales entre otros. En junio del año 2008, la ISO reconoce la versión **PDF** 1.6 como un estándar (ISO 32000-1:2008). El formato **PDF**, según (11), se caracteriza por:

- Multiplataforma: los archivos PDF se pueden ver e imprimir en cualquier plataforma: Mac OS, Microsoft Windows, UNIX y muchas plataformas móviles.

- Extensible: más de 1.800 empresas proveedoras en todo el mundo ofrecen soluciones basadas en PDF, que incluyen creación, plug-in [18] , consultorías, formación y herramientas de asistencia técnica.

- Fiable y robusto: el hecho de que haya más de 250 millones de documentos PDF circulando en la red hoy en día, junto con innumerables archivos PDF en administraciones públicas y negocios

[18] Plug-in: Extensiones para programas.

- Sofisticado en cuanto a la integridad de la información: los archivos PDF tienen el mismo aspecto y muestran la misma información que los archivos originales, como, por ejemplo, texto, dibujos, 3D, gráficos en color, fotos e incluso lógica empresarial, independientemente de la aplicación utilizada para crearlos.

- Capacidad de búsqueda: aprovecha las funciones de búsqueda de texto en documentos y en metadatos.

- Accesible: los documentos PDF colaboran con las tecnologías de asistencia para facilitar el acceso a la información a personas con discapacidades.

Además de las características antes mencionadas se puede agregar que el **PDF** es un formato seguro ya que a partir de la versión **PDF** 1.3 incorporó la **firma digital** y posteriormente cifrado **RC4** y **AES**.

1.8 Firma Digital de PDF utilizando Tarjetas Inteligentes

Cada vez más administraciones y empresas requieren el envío de documentos sobre soporte electrónico. Con el uso aplicaciones para la Firma Digital estas organizaciones pueden enviar a través de Internet documentos importantes en formato PDF, como ofertas de negocios, facturas electrónicas y formularios de una forma rápida y sin que estos sean alterados. Existen múltiples aplicaciones que permiten firmar digitalmente documentos PDF, a continuación algunas características de las más usadas:

- **Adobe Acrobat 9.0**: es la aplicación más completa para edición y creación de documentos en formato **PDF** y permite la creación de este tipo de documento a partir de múltiples formatos. Además se pueden crear formularios en los documentos y darles seguridad a través de la **firma digital** y el cifrado. Se caracteriza por ser una aplicación muy robusta y se integra con el almacén de llaves de Windows, a través del cual puede hacer uso de las **tarjetas inteligentes.** La desventaja está dada porque al ser una aplicación profesional su costo es elevado.

- **Ascertia PDF Sign&Seal**: herramienta de software especializada en **firma digital** y seguridad de documentos en formatos **PDF**, posee su propio visor de documentos y se integra con el almacén de llaves de Windows, a través del cual puede hacer uso de las **tarjetas inteligentes**; incluye **sellado de tiempo** y tiene una interfaz gráfica muy amigable.

- **PortableSigner**: está desarrollado en lenguaje java y permite el uso de certificados en diferentes formatos sin necesidad de importarlos en el sistema operativo. Entre las ventajas de **PortableSigner** están la sencillez de su interfaz de usuario y su característica multiplataforma; y como desventaja, que no incorpora la tecnología de **tarjetas inteligentes** en el proceso de **firma digital**.

- **J4Sign**: es una aplicación desarrollada en lenguaje java especialmente para la Firma Digital utilizando **tarjetas inteligentes**. Es una aplicación multiplataforma y su principal desventaja es que no utiliza otro tipo de certificado.

En resumen, se han analizado varias aplicaciones presentes en el mercado de la seguridad digital para garantizar la autenticidad e integridad de documentos en formato PDF. Basándose en este estudio la solución que se propone en el presente trabajo incluye (ver Figura 7):

- **Firma digital** utilizando **tarjetas inteligentes**.

- **Firma digital** utilizando ficheros certificados[19].

- Sellado de tiempo.

- Soporte para múltiples sistemas operativos.

[19] Los certificados pueden estar en forma de archivos. Las extensiones más comunes son .P12 y .PFX.

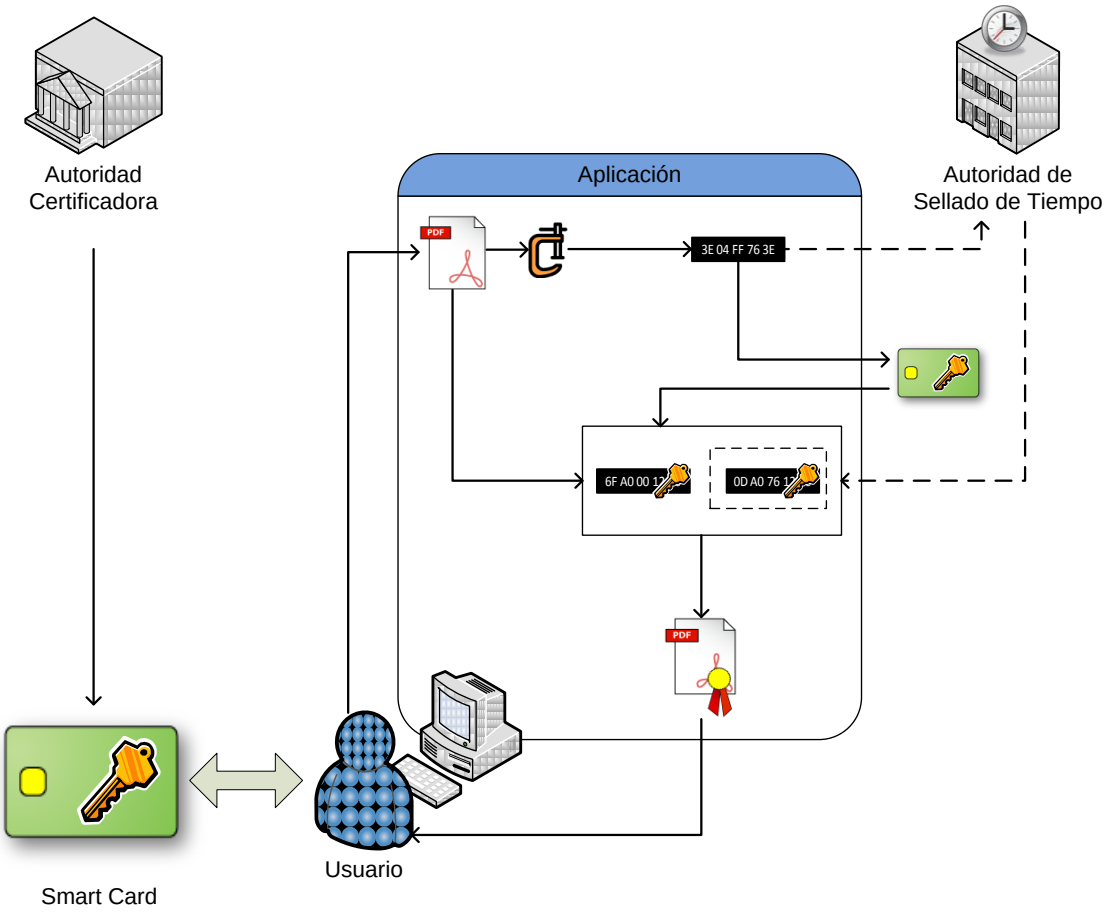

Figura 7 Solución propuesta.

1.9 Conclusiones del capítulo

A partir de la revisión bibliográfica realizada para el desarrollo de este capítulo, se concluye:

1. La criptografía provee los métodos y mecanismos para garantizar la **integridad**, **autenticidad** y **no repudio** de la información.
2. El algoritmo SHA-1/RSA es el estándar de facto para la realización de la **firma digital**.
3. La utilización de dispositivos seguros aumentan la confiabilidad del proceso de **firma digital**.

Capítulo 2: Propuesta de Solución.

2.1 Introducción

La propuesta de solución al problema científico se abordará utilizando la metodología de desarrollo de software XP (Extreme Programming). Esta metodología propone seis fases y el objetivo que se persigue con la elaboración de este capítulo es mostrar la evolución de la solución durante las fases de Exploración y Planificación, además de presentar los diferentes artefactos generados en dichas fases.

2.2 Selección de la metodología de desarrollo

La selección de una metodología de desarrollo de software adecuada, es a veces un factor determinante en el éxito de un proyecto. Aunque no existe una metodología absoluta, algunas se ajustan mejor que otras, a las características y necesidades específicas de los proyectos de desarrollo.

Dentro de las metodologías de desarrollo existen dos grandes grupos, las conocidas Metodologías Tradicionales y las Metodologías Ágiles; según (12), las primeras enfatizan en el uso exhaustivo de documentación durante todo el ciclo de vida del proyecto, mientras que las segundas dan mayor importancia a la capacidad de respuesta a los cambios y a mantener una buena relación con el cliente para llevar al éxito el proyecto.

Las más conocidas de cada grupo son: RUP (Rational Unified Process) y MSF (Microsoft Solution Framework) por las tradicionales y XP (Extreme Programming), Iconix y Scrum por las ágiles. La selección se enmarcó en RUP y XP por ser las metodologías líderes en sus respectivos grupos, además son muy conocidas en la Universidad de las Ciencias Informáticas, por lo tanto cuentan con mayor soporte de documentación y expertos en el tema.

2.2.1 Rational Unified Process (RUP)

RUP es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través del UML (Unified Modeling Language), y trabajo de muchas metodologías utilizadas por los clientes. La versión que se ha estandarizado vio la luz en 1998 y se conoció en sus inicios como Proceso Unificado de Rational 5.0; de ahí las siglas con las que se identifica a este proceso de desarrollo (13).

Características de RUP:

- Está dirigido por Casos de Uso.
- Está centrado en la Arquitectura.
- Es iterativo e incremental.
- Utiliza UML como lenguaje de modelado.
- Puede ser adaptado a las necesidades del proyecto.

Principales elementos:

- Trabajadores (“quién”): Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.
- Actividades (“cómo”): Es una tarea que tiene un propósito claro y es realizada por un trabajador.
- Artefactos (”qué”): Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.
- Flujo de actividades (“cuándo”): Secuencia de actividades realizadas por trabajadores y que producen un resultado de valor observable.

Además define cuatro fases:

- Conceptualización (Concepción o Inicio).
- Elaboración.
- Construcción.
- Transición.

Estableciendo nueve flujos de trabajo presentes a lo largo de las fases mencionadas anteriormente, los flujos son:

1. Modelado del negocio.
2. Requerimientos.

3. Análisis y diseño.
4. Implementación.
5. Instalación.
6. Administración del proyecto.
7. Administración de configuración y cambios.
8. Ambiente.

2.2.1.1 Ventajas

- Ideal para proyectos de mediano a gran alcance y con grandes grupos de desarrollo que además pueden estar distribuidos.
- Funciona bien en proyectos de innovación.
- Es sencillo, ya que sigue los pasos intuitivos necesarios a la hora de desarrollar el software.
- Seguimiento detallado en cada una de las fases.

2.2.1.2 Desventajas

- Altos costos de cambio dados por la planificación a largo plazo y la trazabilidad que existe entre los diferentes artefactos a lo largo de todas las fases.

2.2.2 Extreme Programming (XP)

XP es actualmente un enfoque deliberado y disciplinado para el desarrollo de software. La metodología está diseñada para entregar el software que el cliente necesita, en el momento que lo necesita(14). Además promueve el uso de prácticas para aumentar la productividad del equipo de desarrollo y mejorar la adaptabilidad a los frecuentes cambios dentro del ciclo de vida del proyecto.

Algunas de las prácticas más usadas son:

- Entregas pequeñas y frecuentes.
- Cliente in-situ.
- Diseños simples.

- Integración continua.
- Programación en parejas.
- Desarrollo guiado por pruebas.
- Estándares y refactorización de código

Estas prácticas soportan los cuatro principios de la metodología:

1. Comunicación.
2. Simplicidad.
3. Retroalimentación.
4. Coraje.

2.2.2.1 Ventajas

- Apropiado para entornos volátiles, equipos de desarrollo pequeños (de 2 a 10 desarrolladores) y proyectos de alto riesgo.
- Permite una mejor adaptabilidad a los cambios, que se traduce en una reducción de costos.
- Planificación a corto plazo y más transparente para los clientes, ya que conocen las fechas de entrega de funcionalidades vitales para su negocio.
- Permite tener retroalimentación continua de los usuarios a través de las entregas frecuentes.
- La presión está a lo largo de todo el proyecto y no en una entrega final.

2.2.2.2 Desventajas

- A veces cuesta delimitar el alcance del proyecto con el cliente.

2.2.3 ¿Por qué XP?

Después de un análisis de las características, ventajas y desventajas de ambas metodologías se identificaron ciertos factores que luego potenciaron la elección de XP como metodología a utilizar. Los elementos determinantes fueron:

- El tamaño del grupo de desarrollo, en este caso, de apenas dos personas.
- Cliente presente en el grupo de desarrollo.
- Necesidad de resultados tangibles a corto plazo, entiéndase esto como versiones funcionales de la solución.
- Imposibilidad, para un grupo de desarrollo pequeño, de asumir una metodología robusta, debido a la cantidad excesiva de roles y documentación generada en el ciclo de vida del proyecto.
- Problema a solucionar de complejidad media.

2.3 Metodología XP. Fases

La metodología XP, como metodología ágil, enfatiza en el carácter interactivo e incremental del desarrollo. Según (15), una iteración es un período de una a cuatro semanas, en el cual el cliente selecciona las funcionalidades que desea que se implementen en dicha iteración. Al final de una iteración el cliente puede ejecutar las pruebas funcionales relativas a la iteración. Estas iteraciones tienen lugar a lo largo de las diferentes fases.

Existe una fase de análisis inicial orientada a planificar las iteraciones de desarrollo y cada iteración incluye diseño, codificación y pruebas, sub-fases superpuestas de tal manera que no se separen en el tiempo.

La siguiente figura muestra las fases en las que se subdivide el ciclo de vida de un proyecto de software con XP:

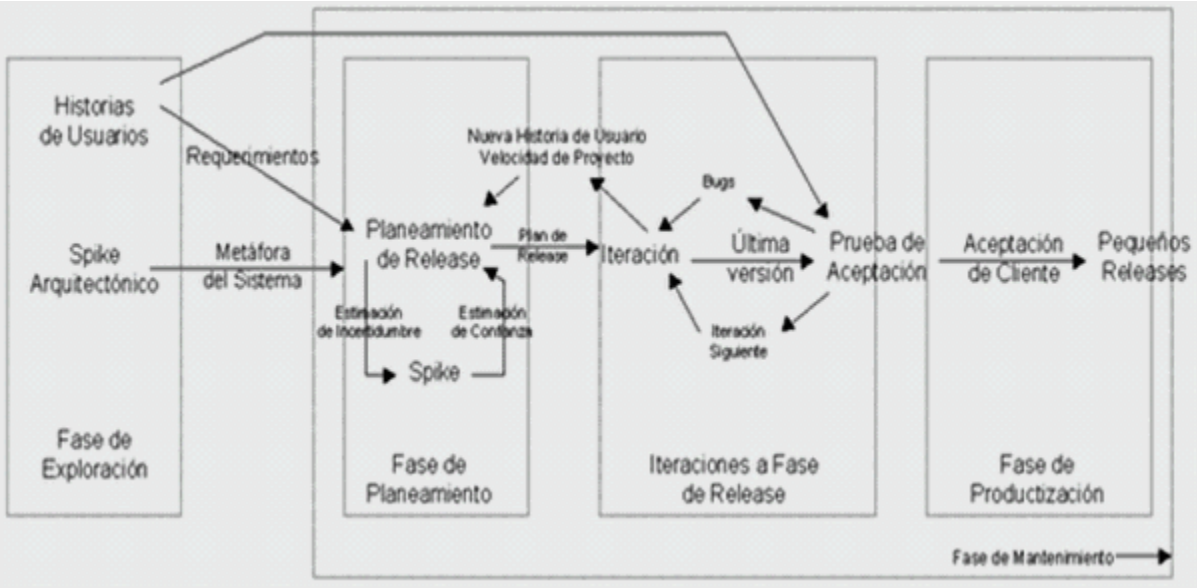

Figura 8 Fases de un proyecto con XP.

A continuación veremos algunos detalles de las fases de Exploración y Planeamiento. El resto de las fases serán tratadas en el próximo capítulo.

2.3.1 Fase de Exploración

En esta fase, los clientes plantean qué necesitan que haga el sistema, y dichas necesidades se recogen en lo que XP define como *Historias de Usuarios*. Las *Historias de Usuarios* (ver tabla 2.1) sirven como lista de requerimientos del sistema; además son utilizadas para estimar el tiempo en la planificación de las liberaciones.

Tabla 2.1: Planilla de Historia de Usuario.

Historia de Usuario	
Número: <Número identificador de la Historia de Usuario>	**Nombre:** <Nombre de la Historia de Usuario>
Usuario:	
Prioridad del Negocio: <Alta, Media, Baja>	**Riesgo de Desarrollo:** <Alta, Media, Baja>
Puntos Estimados: <0,1,2,3>	**Iteración Asignada:** <1, …, n>

Descripción:
Observaciones:

Al mismo tiempo que se definen las *Historias de Usuarios,* el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto; además se exploran las posibilidades de la arquitectura del sistema.

El tiempo invertido en la fase de exploración, toma de unas semanas a unos pocos meses, dependiendo del tamaño del problema en cuestión y el grado de familiarización que hayan alcanzado los miembros del equipo de desarrollo con las tecnologías a utilizar.

2.3.2 Fase de Planeamiento

El propósito de la fase de Planeamiento es que los clientes y programadores se pongan de acuerdo en la fecha de las liberaciones, además los clientes priorizan las *Historias de Usuario* que deberán ser desarrolladas en cada liberación. Con una buena fase de Exploración la fase de Planeamiento debería tomar entre uno y dos días.

En el caso del planeamiento de la primera liberación puede tomar entre dos y seis meses porque es donde se tienen en cuenta los problemas más significativos del negocio, además los programadores estiman cuánto esfuerzo requiere cada historia y a partir de allí se define el cronograma.

Los artefactos que se generan en esta fase son el *Plan de Iteraciones* y el *Plan de Entrega.*

2.4 Propuesta de Solución

Después de haberse precisado algunos conceptos básicos de la metodología XP, específicamente de las fases de Exploración y Planificación, se tratarán los aspectos específicos de la solución propuesta.

2.4.1 Historias de Usuario

Durante la fase de Exploración se definieron las diferentes *Historias de Usuarios*; las tablas a continuación describen cada una de ellas.

Tabla 2.2: Historia de Usuario Gestión de Archivos PDF.

Historia de Usuario	
Número: HU1	**Nombre:** Gestionar Archivos PDF
Usuario: Desarrollador	
Prioridad del Negocio: Alta	**Riesgo de Desarrollo:** Bajo
Puntos Estimados: 1	**Iteración Asignada:** 1
Descripción: Permitirá cargar los archivos en formato PDF que serán firmados digitalmente, seleccionando la ruta del archivo PDF o un directorio contenedor de varios archivos PDF. Además permitirá definir el nombre que se desea tenga el archivo firmado o el directorio con los archivos firmados.	
Observaciones:	

Tabla 2.3: Historia de Usuario Gestión de Certificados Digitales.

Historia de Usuario	
Número: HU2	**Nombre:** Gestión de Certificados Digitales
Usuario: Desarrollador	
Prioridad del Negocio: Alta	**Riesgo de Desarrollo:** Media
Puntos Estimados: 1	**Iteración Asignada:** 1
Descripción: Permitirá cargar el contenedor de la llave privada destinada a la firma digital y el certificado digital correspondiente. El contenedor de llaves podrá ser una tarjeta inteligente o un archivo. Además permitirá visualizar detalles del certificado digital como el propietario, el proveedor y la fecha de validez del mismo.	
Observaciones: Debe ofrecer la opción de introducir la contraseña para acceder a la información del archivo.	

Tabla 2.4: Historia de Usuario Gestión de la Apariencia de la Firma Digital.

Historia de Usuario	
Número: HU3	**Nombre:** Gestión de la Apariencia de la Firma Digital
Usuario: Desarrollador	
Prioridad del Negocio: Media	**Riesgo de Desarrollo:** Media
Puntos Estimados: 1	**Iteración Asignada:** 2
Descripción: Permitirá establecer la visibilidad de firma digital en el documento PDF; en caso de que la firma sea visible se debe especificar en qué páginas del documento estará presente. Además permitirá adicionar una imagen a la firma digital y datos como la razón por la cual se firma el documento y la localización.	
Observaciones:	

Tabla 2.5: Historia de Usuario Gestión del Servicio de Sellado de Tiempo.

Historia de Usuario	
Número: HU4	**Nombre:** Gestión del Servicio de Sellado de Tiempo
Usuario: Desarrollador	
Prioridad del Negocio: Media	**Riesgo de Desarrollo:** Media
Puntos Estimados: 2	**Iteración Asignada:** 2
Descripción: Dará la posibilidad de incluir sellado de tiempo a los documentos firmados, y en caso de que sea incluida esta opción, se podrá elegir el servidor de sellado de tiempo a utilizar y se deberá especificar el usuario y la contraseña (en caso de que sea necesario) para acceder a este servicio a través de Internet.	
Observaciones:	

Tabla 2.6: Historia de Usuario Firmar documento PDF.

Historia de Usuario	
Número: HU5	**Nombre:** Firmar documento PDF
Usuario: Desarrollador	
Prioridad del Negocio: Alta	**Riesgo de Desarrollo:** Media
Puntos Estimados: 2	**Iteración Asignada:** 1
Descripción: Firmará digitalmente un documento PDF utilizando un certificado digital que puede estar almacenado en un archivo o en una tarjeta inteligente y como resultado de la operación creará un nuevo documento PDF firmado con el certificado seleccionado.	
Observaciones:	

Tabla 2.7: Historia de Usuario Gestión de la Configuración.

Historia de Usuario	
Número: HU6	**Nombre:** Gestión de la Configuración
Usuario: Desarrollador	
Prioridad del Negocio: Baja	**Riesgo de Desarrollo:** Medio
Puntos Estimados: 2	**Iteración Asignada:** 2
Descripción: Dará la posibilidad de configurar opciones adicionales como la configuración del proxy de red, administración de la lista de servidores de sellado de tiempo y selección del proveedor de tarjetas inteligentes.	
Observaciones:	

2.4.2 Requerimientos no funcionales

Además de las funcionalidades descritas en las historias de usuario se tienen en cuenta los requerimientos no funcionales o propiedades que debe cumplir el producto. Estos requerimientos estuvieron enfocados básicamente al cumplimiento de estándares y a la independencia del sistema operativo.

Requerimientos mínimos de Hardware:

- Microprocesador Intel® Pentium IV o semejante, con velocidad de procesamiento de 2.0 GHz o superior.
- Memoria RAM de 256 Mb.
- Espacio en disco de 6 Mb.

Requerimientos de Software:

- Para el correcto funcionamiento de la aplicación se requiere el uso de la maquina virtual de java JDK versión 6.0 o superior.
- El sistema debe ser independiente de sistema operativo y ser completamente funcional en los principales sistemas operativos existentes en el mercado: Microsoft Windows, Mac OS y Linux.

Requerimientos de Diseño e Implementación:

- El sistema debe ser diseñado para interactuar con dispositivos criptográficos compatibles con el estándar PKCS#11[20] [21].
- El sistema debe ser diseñado para interactuar con archivos compatibles con el estándar PKCS#12[22].

[20] PKCS (Public Key Cryptographic Standars): Grupo de estándares de criptografía de llave pública desarrollados por RSA Security Labs.
[21] PKCS#11 Define un API llamada Cryptoki (cryptographic token interface) para dispositivos que guardan información criptográfica y realizan funciones criptográficas. El objetivo es abstraer los detalles del dispositivo criptográfico (token) y ofrecer a las aplicaciones un modo común de ver los tokens.
[22] PKCS#12 Este estándar describe la sintaxis que deben tener los datos de identidad personal (llaves privadas, certificados, información secreta y otros) al ser transportados.

2.4.3 Herramientas y tecnologías

Otro de los puntos que define la metodología XP para la fase de Exploración es la selección de las herramientas y tecnologías a utilizar durante el periodo de desarrollo. Esta selección se realiza partiendo de la experiencia del equipo de desarrollo y de las necesidades específicas del cliente.

2.4.3.1 Java como lenguaje de programación

La selección del lenguaje de programación estuvo determinada por la necesidad de desarrollar una solución sobre plataformas abiertas, siguiendo las nuevas líneas de desarrollo del Centro de Identificación y Seguridad, y que fuese además independiente del sistema operativo. Por estas razones se seleccionó Java como lenguaje de programación.

Java, a diferencia de otros lenguajes, fue diseñado pensando en su seguridad desde el principio. Desde la versión 1.1 de la maquina virtual de java se incorporó al lenguaje el framework JCA (Java Cryptography Architecture) para acceder a las principales funciones criptográficas como funciones hash y firma digital; y luego como paquete opcional JCE (Java Cryptography Extension) que provee un framework e implementaciones para algoritmos de cifrado, generación de llaves e intercambio de llaves. El soporte para cifrado incluye cifrado simétrico, asimétrico, por bloques y por flujo.

Estas y otras características bien conocidas del lenguaje, como la orientación a objetos, la recolección automática de basura y la integración con un sinnúmero de plataformas y tecnologías hacen de Java el lenguaje ideal para implementar la solución.

2.4.3.2 Netbeans IDE 6.5 como entorno integrado de desarrollo

Netbeans IDE 6.5 es un entorno integrado de desarrollo gratuito, de código abierto para desarrolladores de software. Incluye todas las herramientas necesarias para crear aplicaciones profesionales para el escritorio, la empresa, la web y equipos móviles con el lenguaje Java. Netbeans IDE es fácil de instalar y se ejecuta en varias plataformas incluyendo Windows, Linux, Mac OS X y Solaris. Además contiene un diseñador de interfaces de Swing que permite la creación rápida y de manera visual de las interfaces de usuario para entornos de escritorio.

2.4.3.3 Librería de clases para manejo de documentos PDF

iText 2.1.4 es una librería de clases gratuita y de código abierto, que brinda funcionalidades para la creación y edición de manera profesional de documentos **PDF** utilizando lenguaje Java. Entre sus principales funcionalidades están:

1. Exponer documentos **PDF** en los navegadores web.
2. Generación dinámica de documentos a partir de XML o bases de datos.
3. Uso de las características de interactividad del formato **PDF**.
4. Adición de marcadores, números de página, marcas de agua, etc.
5. Dividir, concatenar y manipular páginas de documentos **PDF**.
6. Manejo de datos de formularios.
7. Adición y verificación de campos de firma digital a los documentos **PDF**.
8. Protección de documentos **PDF** a través de cifrado simétrico.

Este proyecto brinda un amplio soporte mediante fórums mantenidos por la comunidad de desarrollo y sitios oficiales con documentación en línea.

2.4.3.4 Proveedor criptográfico de Bouncy Castle

Los proveedores criptográficos no son más que librerías que implementan algoritmos criptográficos y pueden ser incorporados al código de la aplicación, de manera transparente, para ser utilizados de la misma forma que los predefinidos por JCA y JCE.

El proveedor criptográfico de Bouncy Castle, es uno de los más utilizados por la comunidad de criptografía para el desarrollo de aplicaciones con elementos de seguridad digital. Se puede obtener de manera gratuita, su código es abierto y sus APIs criptográficas para Java consisten en:

1. Un proveedor criptográfico para JCA y JCE.
2. Una implementación clara de JCE 1.2.1.
3. Librerías para el manejo de objetos con codificación ASN.1.[23]
4. Generadores para certificados X.509 versiones 1 y 3, CRLs[24] versión 2 y archivos PKCS12.
5. Generadores de atributos de certificados X.509 versión 2.

[23] ASN.1 (Abstract Syntax Notation 1) estándar desarrollado por ISO/ITU-T y su objetivo es proveer una notación estándar para almacenar información.
[24] CLR (Certificates Revocation List) lista con los números de serie de los certificados revocados por una entidad certificadora.

6. Generadores /Procesadores para OCSP[25] (RFC 2560).

7. Generadores /Procesadores para TSP[26] (RFC 3161).

2.4.3.5 Proveedor criptográfico de PKCS#11 de Sun

El proveedor de PKCS # 11 de Sun, en contraste con la mayoría de los otros proveedores, no implementa algoritmos criptográficos en sí. En lugar de ello, actúa como un puente entre las APIs JCA y JCE de Java y las APIs criptográficas de PKCS#11 nativas, traduciendo las llamadas entre los dos. Esto significa que las aplicaciones Java utilizando las APIs JCA y JCE pueden, sin modificación alguna, tomar ventaja de los algoritmos ofrecidos por las implementaciones de PKCS # 11, como, por ejemplo:

- Tarjetas Inteligentes,
- Aceleradores criptográficos de hardware, e
- Implementaciones de software de alto rendimiento.

Tenga en cuenta que J2SE[27] sólo facilita el acceso a implementaciones PKCS#11 nativas, y no que incluye una implementación de PKCS#11. Sin embargo, los dispositivos criptográficos como **tarjetas inteligentes** y aceleradores de hardware a menudo vienen con software que incluye una implementación del PKCS#11 que usted necesita instalar y configurar de acuerdo a las instrucciones del fabricante(16).

2.4.4 Metáfora del Sistema

Después de haberse definido las funcionalidades que el sistema debe cumplir, los requerimientos no funcionales y las herramientas de desarrollo, el equipo procede a la creación de la *Metáfora*. Está es una breve descripción de cómo debe funcionar el sistema en su totalidad. La *Metáfora* guía todo el desarrollo como una gran *Historia de Usuario* ayudando al equipo a entender los elementos básicos del sistema y sus relaciones.

[25] OCSP (Online Certificate Status Protocol) protocolo que permite determinar si un certificado está revocado o no, a través de un servicio expuesto por la autoridad certificadora que lo emitió.
[26] TSP (Time Stamp Protocol) Protocolo de intercambio de información entre aplicaciones y la autoridad de sellado de tiempo. Define la estructura de los pedidos y respuestas de la autoridad de sellado de tiempo.
[27] J2SE (Java 2 Standar Edition) es la plataforma de desarrollo de aplicaciones de entorno de escritorio con lenguaje Java.

Tabla 2.8: Metáfora del Sistema.

Metáfora
Para firmar uno o varios documentos PDF se seleccionará el documento o la carpeta con los documentos en formato PDF a firmar; luego el nombre que tendrá el documento firmado o la carpeta que contendrá los documentos firmados. Será necesario seleccionar el certificado digital para realizar la firma digital e introducir la contraseña para acceder a este contenido. Después de estas acciones se podrá realizar la firma digital. Además se podrá definir opciones adicionales como apariencia de la firma y sellado de tiempo. *Para realizar las opciones de sellado de tiempo será necesario el uso de un servidor de sellado de tiempo, servicio externo a la solución.*

2.4.5 Estimación de Tiempo

Concluida la fase de Exploración los programadores estiman el tiempo que necesitan para desarrollar cada Historia de Usuario, este valor se expresa en semanas. Una Historia de Usuario no debe desarrollarse en menos de una, ni en más de dos semanas; en otro caso, será necesario acoplar o dividir las Historias de Usuarios. Como se ha dicho anteriormente, este valor es estimado y se irá acercando a la realidad con el transcurso de las iteraciones.

Tabla 2.9: Estimación de tiempo.

Historia de Usuario	Estimación
Gestionar Archivos PDF	1
Gestionar Certificados Digitales	2
Gestionar Apariencia de la Firma Digital	1
Gestionar Servicio de Sellado de Tiempo	2
Firmar documento PDF	2
Gestionar Configuración	1
Total	**9**

2.4.6 Plan de Iteraciones

Luego a partir de las prioridades definidas por el cliente se crea el *Plan de Iteraciones*. Generalmente las *Historias de Usuarios* con mayor prioridad son asignadas a las primeras iteraciones, y las de mediana y menor prioridad a las iteraciones posteriores. En la presente solución se han identificado siete *Historias de Usuarios* y se definieron dos iteraciones; la primera de seis semanas de duración y la segunda de cuatro semanas (ver tabla 2.11).

Tabla 2.10: Plan de Iteraciones.

Iteración	No. HU	Historia de Usuario	Duración Estimada
Iteración 1	HU1	Gestionar Archivos PDF	5 semanas
	HU2	Gestionar Certificados Digitales	
	HU5	Firmar documento PDF	
Iteración 2	HU3	Gestionar Apariencia de la Firma Digital	4 semanas
	HU4	Gestionar Servicio de Sellado de Tiempo	
	HU6	Gestionar Configuración	

2.4.7 Plan de Entregas

Para finalizar la fase de Planeamiento se acuerda con el cliente el *Plan de Entregas* que define las fechas en que serán liberadas las versiones funcionales del producto. Este artefacto cumple con el principio de la metodología XP de las "liberaciones frecuentes", generalmente asociadas al fin de un grupo de iteraciones. A continuación se muestra el *Plan de Entrega*:

Tabla 2.11 Plan de Entregas.

Entregable	Fin Iteración 1 2009/04/13	Fin Iteración 2 2009/05/11
PDFSigner.jar	**Versión 0.1**	**Versión 1.0**

2.5 Conclusiones del capítulo

Luego de terminadas las fases de Exploración y Planificación de la solución propuesta, se concluye:

1. Se definieron seis *Historias de Usuario* siendo las más importantes:
    a. Gestionar Certificados Digitales.
    b. Firmar Documentos PDF.
    c. Gestionar Servicio de Sellado de Tiempo.
2. Se seleccionaron las herramientas y tecnologías de código abierto para el desarrollo de la solución.
3. Se definieron dos iteraciones con el objetivo de planificar el trabajo del equipo de desarrollo.

Capítulo 3: Implementación de la Solución

3.1 Introducción

Seguido de la fase de planificación XP define las fases “Iteraciones a primera Liberación” y “Producción”. En la planificación se definieron las iteraciones y en cada iteración se diseñan, se prueban y codifican cada una de las historias de usuario. El objetivo que se persigue con la elaboración de este capítulo es mostrar la evolución de la solución durante las fases de “Iteraciones a primera Liberación” y “Producción”.

3.2 Iteraciones a primera liberación

En la fase “Iteraciones a primera liberación” es donde se da cumplimento al “Plan de Iteraciones” (ver epígrafe 2.4.6). En cada iteración se desarrollan las sub-fases de “Diseño”, realización de “Pruebas Unitarias”, “Codificación de la Solución” y “Refactorización”. Dada la naturaleza ágil de XP la sub-fase más importante del ciclo de vida del proyecto es la “Codificación de la Solución”. Al finalizar la fase de “Iteraciones a Primera Liberación” el cliente estará apto para realizar las pruebas de aceptación.

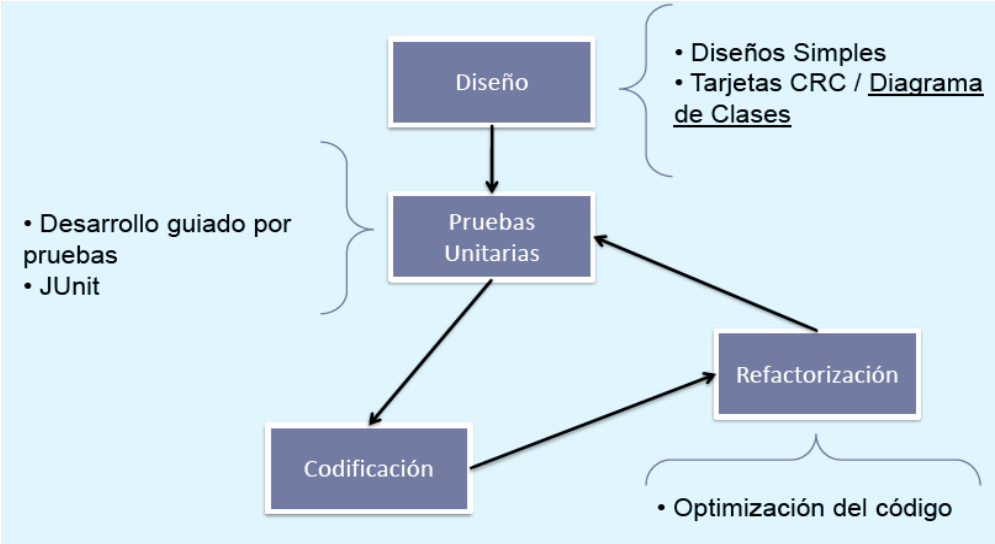

Figura 9 Sub-fases de la fase Iteraciones a primera Liberación.

3.2.1 Tareas de la ingeniería

Antes de comenzar a codificar la solución es necesario saber qué codificar. Las historias de usuarios no ofrecen el nivel de detalles requerido para llevar a cabo esta acometida, es por eso que son divididas en tareas de la ingeniería. Una historia de usuario generalmente se divide en más de una tarea de la ingeniería y a partir de estas tareas comienza el ciclo de la fase de Iteraciones a primera liberación.

Según el plan de iteraciones (ver epígrafe 2.4.6) las historias de usuario se agruparon en dos iteraciones. A continuación se muestran las tareas de la ingeniería derivadas de cada historia de usuario por iteración.

3.2.1.1 Tareas de la ingeniería Iteración 1.

HU1 Gestionar Archivos PDF:

Tabla 3.1 HU1_T1

TAREA	
Número de Tarea: HU1_T1	**Historia de Usuario:** Gestionar Archivos PDF
Nombre: Diseño de Interfaz para Gestión de Archivos PDF	
Tipo: Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Responsable: Roberto Quiñones	
Descripción: La interfaz debe contar con campos de texto para entrar la dirección del documento PDF y botones que muestren un selector de ficheros para seleccionar el documento visualmente.	

Tabla 3.2: HU1_T2

TAREA	
Número de Tarea: HU1_T2	**Historia de Usuario:** Gestionar Archivos PDF
Nombre: Cargar Archivos PDF	
Tipo: Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Responsable: Roberto Quiñones	
Descripción: El usuario podrá introducir en un campo de texto la ruta del documento o directorio	

contenedor de varios documentos PDF que desee firmar, o seleccionarlos de manera visual. El sistema debe validar que la ruta del documento o directorio sea correcta, en caso contrario debe mostrar un mensaje de error informando al usuario.

Tabla 3.3: HU1_T3

TAREA	
Número de Tarea: HU1_T3	**Historia de Usuario:** Gestionar Archivos PDF
Nombre: Salvar Archivos PDF	
Tipo: Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Responsable: Roberto Quiñones	
Descripción: El usuario podrá introducir en un campo de texto la ruta donde desea guardar el documento o los documentos PDF firmados, o seleccionarlos de manera visual. El sistema debe validar que la ruta del documento o directorio sea correcta, en caso contrario debe mostrar un mensaje de error informando al usuario.	

HU2 Gestionar Certificados Digitales:

Tabla 3.4 HU2_T1

TAREA	
Número de Tarea: HU2_T1	**Historia de Usuario:** Gestionar Certificados Digitales
Nombre: Diseño de Interfaz para Gestión de Certificados Digitales	
Tipo: Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Responsable: Roberto Quiñones	
Descripción: La interfaz debe contar con un campo de selección que contendrá los certificados existentes y un botón para mostrar los detalles del certificado seleccionado.	

Tabla 3. 5 HU2_T2

TAREA	
Número de Tarea: HU2_T2	**Historia de Usuario:** Gestionar Certificados Digitales
Nombre: Cargar Certificados Digitales	
Tipo: Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Responsable: Roberto Quiñones	
Descripción: El usuario podrá introducir en un campo de texto la ruta del archivo contenedor del certificado o seleccionarlo de manera visual, será necesario la introducción de la contraseña para acceder al mismo. El usuario podrá ver los detalles del certificado. El sistema debe validar que la ruta y la contraseña sean correctas, en caso contrario debe mostrar un mensaje de error informando al usuario. Además debe soportar las extensiones de archivo .pfx y .p12	

Tabla 3.6 HU2_T3

TAREA	
Número de Tarea: HU2_T3	**Historia de Usuario:** Gestionar Certificados Digitales
Nombre: Cargar módulo PKCS#11 asociado a una Tarjeta Inteligente	
Tipo: Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Responsable: Roberto Quiñones	
Descripción: El usuario podrá introducir en un campo de texto la ruta del módulo PKCS#11 o seleccionarlo de manera visual, será necesario la introducción del pin de acceso a la Tarjeta Inteligente asociada. El sistema debe validar que la ruta y el pin sean correctos y cargar los certificados contenidos en la Tarjeta Inteligente, en caso contrario debe mostrar un mensaje de error informando al usuario. Además debe soportar las extensiones de archivo .dll y .so.	

HU5 Firmar documento PDF:

Tabla 3 7 HU5_T1

TAREA	
Número de Tarea: HU5_T1	**Historia de Usuario:** Firmar documento PDF
Nombre: Firmar documento PDF	
Tipo: Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Responsable: Roberto Quiñones	
Descripción: Se debe cargar el documento PDF a partir de la ruta del mismo, crear el campo de firma digital que será incluido en un nuevo documento. Para realizar esta operación será necesario haber seleccionado un certificado digital. La configuración de la apariencia de la firma digital y el servicio de sellado de tiempo son opcionales.	

Tabla 3.8 HU5_T2

TAREA	
Número de Tarea: HU5_T2	**Historia de Usuario:** Firmar documento PDF
Nombre: Firmar múltiples documento PDF	
Tipo: Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Responsable: Roberto Quiñones	
Descripción: Se deben cargar los documentos PDF a partir de la ruta del directorio contenedor, y para cada documento, realizar el procedimiento descrito en HU5_T1. Se debe utilizar concurrencia para agilizar el proceso.	

3.2.1.2 Tareas de la ingeniería Iteración 2.

HU3 Gestionar Apariencia de la Firma Digital:

Tabla 3.9 HU3_T1

TAREA	
Número de Tarea: HU3_T1	**Historia de Usuario:** Gestionar Apariencia de la Firma Digital
Nombre: Diseño de Interfaz para Gestionar Apariencia de la Firma Digital.	
Tipo: Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Responsable: Roberto Quiñones	
Descripción: La interfaz debe contar con campos de texto para la razón, la localización de la firma digital y el correo electrónico del usuario firmante. Además debe contar con campos para activar o desactivar las opciones de visibilidad de la firma digital y las opciones de certificación de documentos. Opciones de visibilidad: 1. Posición de la página donde se verá el campo de la firma digital: a. Arriba e Izquierda b. Arriba y Derecha c. Abajo e Izquierda d. Abajo y Derecha 2. Página donde aparecerá el campo de la firma digital: a. Primera página b. Última página	

HU4 Gestionar Servicio de Sellado de Tiempo

Tabla 3.10 HU4_T1

TAREA	
Número de Tarea: HU4_T1	**Historia de Usuario:** Gestionar Servicio de Sellado de Tiempo.
Nombre: Diseño de Interfaz para Gestionar Servicio de Sellado de Tiempo.	

Tipo: Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Responsable: Roberto Quiñones	
Descripción: La interfaz deberá contar con un campo que permita habilitar o deshabilitar las opciones de Sellado de tiempo. Opciones de sellado de tiempo: 1. Servidores - campo de selección. a. Usuario - campo de texto. b. Contraseña - campo de texto.	

Tabla 3.11 HU4_T2

TAREA	
Número de Tarea: HU4_T2	**Historia de Usuario:** Gestionar Servicio de Sellado de Tiempo.
Nombre: Gestionar Servidores de Sellado de Tiempo.	
Tipo: Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Responsable: Roberto Quiñones	
Descripción: Deberá agregar, eliminar o modificar un servidor de sellado de tiempo previamente seleccionado en una lista de servidores. De cada servidor se debe especificar: 1. Nombre del servidor 2. Dirección URL El sistema debe validar que no se adicionen servidores con igual nombre ni dirección URL, en caso contrario mostrará un mensaje de error informando al usuario.	

HU6 Gestionar Configuración:

Tabla 3.12 HU6_T1

TAREA	
Número de Tarea: HU6_T1	**Historia de Usuario:** Gestionar Configuración
Nombre: Diseño de interfaz para Gestionar la Configuración.	
Tipo: Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Responsable: Roberto Quiñones	
Descripción: La interfaz deberá contar con las siguientes opciones de configuración: 1. Certificados (ver tareas HU2_T2 y HU2_T3) 2. Apariencia de la Firma Digital (ver tarea HU3_T1) 3. Servidores de Sellado de tiempo (ver tarea HU4_T2) 4. Conexión de Red a. Proxy – campo de texto. b. Puerto – campo de texto. c. Usuario – campo de texto. d. Contraseña – campo de texto. 5. General a. Lenguaje – campo de selección. b. Sufijo – campo de texto. c. Hilos para firma de múltiples documentos – campo de selección.	

Tabla 3.13 HU6_T2

TAREA	
Número de Tarea: HU6_T2	**Historia de Usuario:** Gestionar Configuración
Nombre: Salvar Configuración	
Tipo: Desarrollo	**Puntos Estimados:**

Fecha Inicio:	**Fecha Fin:**
Responsable: Roberto Quiñones	
Descripción: Se debe guardar los datos de configuración recogidos en los campos descritos en HU7_T1, en archivos .properties, en caso que los archivos no existan deberán ser creados.	

Tabla 3.14 HU6_T3

TAREA	
Número de Tarea: HU6_T3	**Historia de Usuario:** Gestionar Configuración
Nombre: Cargar Configuración	
Tipo: Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Responsable: Roberto Quiñones	
Descripción: Se deben cargar los datos de configuración, almacenados en archivos .properties, en los campos descritos en HU7_T1. En caso que los archivos no existan se debe cargar una configuración preestablecida.	

Tabla 3.15 HU6_T4

TAREA	
Número de Tarea: HU7_T4	**Historia de Usuario:** Gestionar Configuración
Nombre: Cargar Configuración por Defecto	
Tipo: Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Responsable: Roberto Quiñones	
Descripción: Se deben cargar los datos de configuración preestablecidos, en los campos descritos en HU6_T1.	

3.3 Diseño

Una vez desglosadas las historias de usuario en tareas, algunos miembros del equipo de desarrollo comienzan a generar ideas de cómo ejecutarlas. Estas ideas se unifican en las sesiones de diseño previas a cada iteración. En estas sesiones, XP propone la puesta en práctica de ciertos principios, descritos a continuación, para garantizar la agilidad en el proceso de desarrollo.

Según (14), estos principios son:

- **Simplicidad**: Un diseño simple siempre se termina más rápido y es más fácil de entender que uno complejo.
- **Uso de tarjetas CRC**: (clase, responsabilidad, colaboración), estas tarjetas son manejadas por el equipo de desarrollo durante la codificación de la solución; generalmente cada tarjeta representa una clase diferente en la codificación y tienen como ventaja que todo el equipo contribuye a la elaboración del diseño de la solución.
- **No adicionar funcionalidades tempranamente**: Mantener el sistema lo más separado de las funcionalidades extras que no sean imprescindibles. Solo el 10% de las funcionalidades extras son utilizadas y hacen perder el 90% del tiempo.

El diseño de la solución se realizó por iteraciones, agregando solo las funcionalidades necesarias en cada iteración. Además se sustituyeron las tarjetas CRC por diagramas de clases de UML, que aunque algunos autores no lo aconsejan, fue posible dado que la complejidad del problema a resolver no implicaba la elaboración de diagramas complejos. De esta manera se dio cumplimiento a los principios antes mencionados.

3.3.1 Descripción del diseño

La solución está estructurada en tres paquetes que dividen la lógica del negocio de la presentación y la configuración, respondiendo a una arquitectura n capas. Esto permite cierto grado de independencia entre las partes de la aplicación, característica que la convierten en una solución escalable y de fácil mantenimiento. A su vez los paquetes se dividen en sub-paquetes que agrupan las clases atendiendo a sus funcionalidades (ver diagrama). A continuación se describen cada uno de ellos:

uci.pdfsigner.core: este paquete contiene todas las clases que dan solución al problema agrupadas en los siguientes sub-paquetes:

- **uci.pdfsigner.core.signer:** agrupa las clases encargadas de la firma digital y verificación de los documentos PDF.
- **uci.pdfsigner.core.certificates:** agrupa las clases para el manejo de almacenes de llaves y certificados digitales.
- **uci.pdfsigner.core.timestamp:** agrupa las clases destinadas a la interacción de con los servidores de sellado de tiempo.

uci.pdfsigner.ui: este paquete contiene las clases de la capa de presentación entre formularios y filtros de archivos.

uci.pdfsigner.common.configuration: contiene las clases que manejan la configuración de la aplicación.

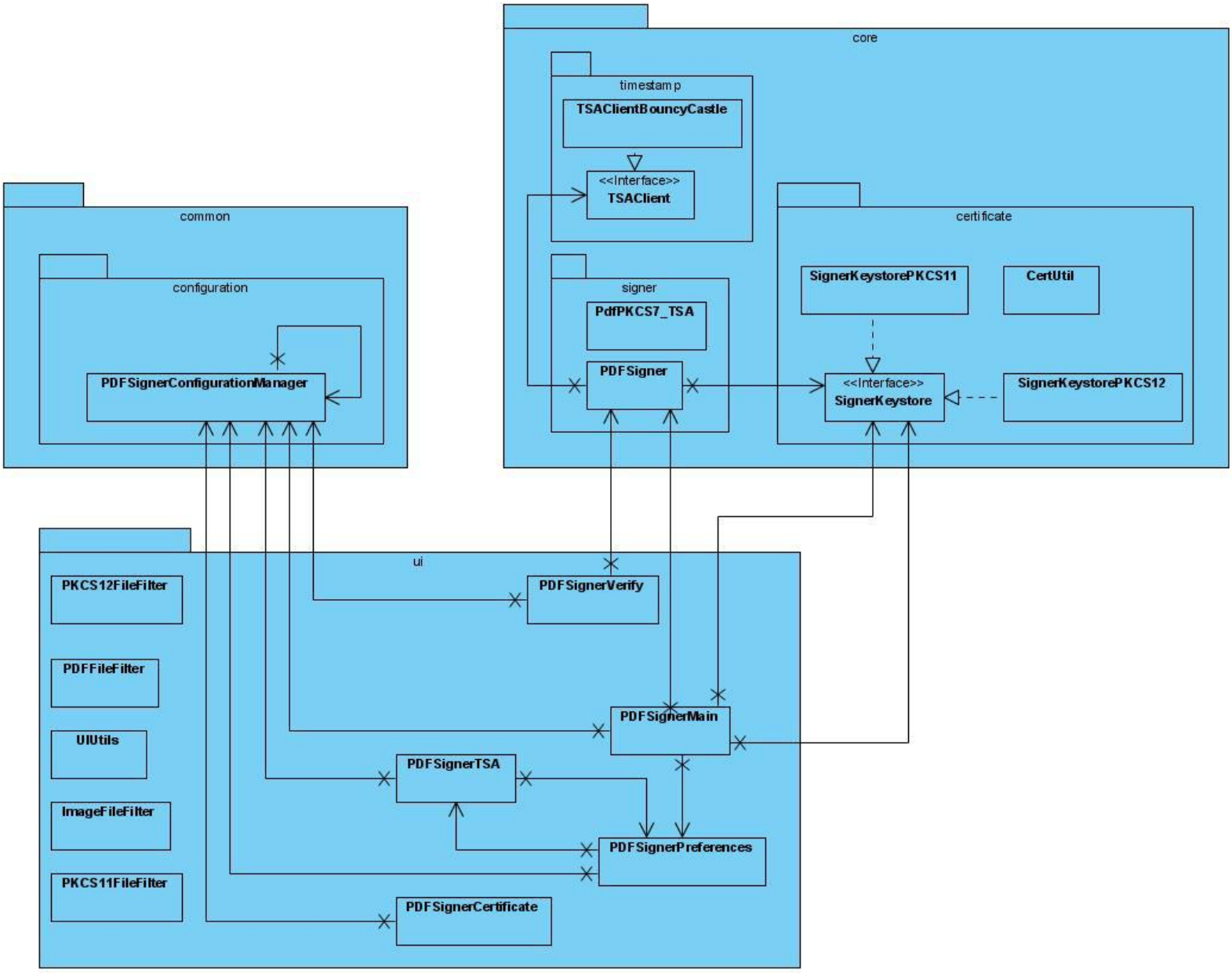

Figura 10 Diagrama de clases

3.4 Pruebas Unitarias y Desarrollo Guiado por Pruebas.

Con el fin de entregar valor lo antes posible, el equipo de desarrollo trabajó solo en los pedidos del cliente, refinando el diseño y el código continuamente para lograr un sistema limpio y capaz de evolucionar. El perfeccionamiento de código sólo fue posible a través de un grupo de pruebas unitarias automatizadas que aseguraron la ejecución correcta del sistema en todo el período de desarrollo.

Las pruebas de unidad se realizaron a través de las librerías JUnit versión 4.5 integradas al IDE. Se construyó un caso de prueba para cada clase de la lógica del negocio y la configuración, con soporte de prueba para los métodos críticos de cada clase. De esta manera se logró la disminución del tiempo en el ciclo compilación-ejecución, permitiendo al equipo de desarrollo concentrarse en la codificación de la solución.

3.5 Codificación

Después de realizar las pruebas de unidad, se procede a la creación del código que cumpla con dichas pruebas, esto asegura una codificación clara y probada desde los primeros momentos del desarrollo.

3.5.1 Descripción de las clases principales

Tabla 3.16 uci.pdfsigner.core.signer.PDFSigner

Paquete contenedor	uci.pdfsigner.core.signer
Super clase	java.lang.Object
Nombre	PDFSigner
Descripción	
Clase encargada de la firma digital y verificación de documentos PDF.	
Atributo	**Tipo**
sks	SignerKeystore
tsaClient	TSAClient
reason	String
location	String
contact	String

signatureVisibility	boolean
useSignImg	boolean
signatureImagePath	String
certificationLevel	int
signaturePosition	int
signaturePage	int

Métodos	
Nombre	PDFSigner(SignerKeystore arg0, TSAClient arg1)
Descripción	Constructor de la clase. Recibe en arg0 un objeto de tipo SignerKeystore que contiene el par de llaves necesarias para la operación de firmado y en arg1 un objeto de tipo TSAClient para la interacción con el servidor de sellado de tiempo.
Nombre	sign(String arg0, String arg1, String arg2)
Descripción	Método encargado de firmar digitalmente un archivo PDF. Recibe en arg0 la ruta del documento PDF que se desea firmar, en arg1 la ruta del nuevo documento firmado, y en arg2 el alias correspondiente al certificado que se utilizará para firmar el documento.
Nombre	sign(String arg0, String arg1, TSAClient arg2, String arg3)
Descripción	Método encargado de firmar digitalmente un archivo PDF. Recibe en arg0 la ruta del documento PDF que se desea firmar, en arg1 la ruta del nuevo documento firmado, en arg2 un objeto de tipo TSAClient para la interacción con el servidor de sellado de tiempo, y en arg3 el alias correspondiente al certificado que se utilizará para firmar el documento.
Nombre	verifySignature(String arg0)
Descripción	Método encargado de verificar la validez de la firma digital de un documento PDF. Recibe en arg0 la ruta del documento PDF a verificar.
Nombre	setAppearance(PdfSignatureAppearance arg0, PdfReader arg1, String arg2)
Descripción	Método encargado de manejar la apariencia de la firma digital, visibilidad, posición. Recibe en arg0 un objeto de tipo PdfSignatureAppearance a través del cual gestiona la apariencia de la firma digital, en arg1 un objeto de tipo PdfReader para acceder a la página donde estará visible el campo de la firma digital en el documento PDF firmado, y en arg2 el alias correspondiente al certificado que se utilizará para firmar el documento.
Nombre	getSignatureContainer(PdfReader arg0, int arg1, int arg2)

Descripción	Método encargado de calcular el tamaño y la posición del rectángulo contenedor de la firma digital, en una página específica del documento. Recibe en arg0 un objeto de tipo PdfReader para acceder a la página donde estará visible el campo de la firma digital en el documento PDF firmado, en arg1 el número de la página, y en arg2 el número de la posición.
Nombre	genPKCS7Signature(InputStream arg0, TSAClient arg1, String arg2)
Descripción	Método encargado de generar la secuencia de bytes correspondiente al resultado de la operación de firma digital del contenido del documento PDF. Recibe en arg0 la el contenido del documento al cual se la puede aplicar la función resumen, en arg1 un objeto de tipo TSAClient para la interacción con el servidor de sellado de tiempo, y en arg2 el alias correspondiente al certificado que se utilizará para firmar el documento.

Tabla 3 17 uci.pdfsigner.core.signer.PdfPKCS7_TSA

Paquete contenedor	uci.pdfsigner.core.signer
Super clase	com.lowagie.text.pdf.PdfPKCS7
Nombre	PdfPKCS7_TSA
Descripción	
Clase encargada de realizar el proceso de firma digital y verificación de firma PKCS#7. Redefine el método getEncodedPKCS7() de la clase com.lowagie.text.pdf.PdfPKCS7 para adicionar soporte de sellado de tiempo.	
Métodos	
Nombre	PdfPKCS7_TSA(PrivateKey arg0, Certificate[ ] arg1, CRL[] arg2, String arg3, String arg4, boolean arg5)
Descripción	Constructor de la clase. Recibe en arg0 la llave privada, en arg1 la cadena de certificados asociada a la llave privada, en arg2 las listas de revocación de certificados, en arg3 el nombre del algoritmo de hash, en arg4 el nombre del proveedor criptográfico, y en arg5 si existen o no los datos de la firma RSA.
Nombre	getEncodedPKCS7(byte[ ] arg0, Calendar arg1, TSAClient arg2)
Descripción	Método encargado de construir el empaquetado PKCS#7. Recibe en arg0 el campo authenticatedAttributes si existe, en arg1 al fecha de firma en el campo authenticatedAttributes si existe, y en arg2 un objeto de tipo TSAClient para la interacción con el servidor de sellado de tiempo.
Nombre	getEncodedPKCS7()
Descripción	Método redefinido para agregar soporte de sellado de tiempo. Utiliza el método

	getEncodedPKCS7(byte[ ] arg0, Calendar arg1, TSAClient arg2).
Nombre	buildUnauthenticatedAttributes(byte[ ] arg0)
Descripción	Método encargado de transformar la secuencia de bytes del token de sellado de tiempo en un objeto ASN.1 para ser incluido en el empaquetado PKCS#7. Recibe en arg0 la secuencia de bytes de token de sellado de tiempo.

Tabla 3.18 uci.pdfsigner.core.certificate.SignerkeyStorePKCS11

Paquete contenedor	uci.pdfsigner.core.certificate
Super clase	SignerkeyStore
Nombre	SignerkeyStorePKCS11
Descripción	
Clase encargada de abstraer el acceso al almacén de llaves, alojado en dispositivos criptográficos como HSM y Smart Card compatibles con el estándar PKCS#11, que se utiliza en el proceso de firma digital. Implementa los métodos de la interfaz SignerkeyStore y utiliza el proveedor criptográfico sun.security.pkcs11.SunPKCS11.	
Atributo	**Tipo**
provider	Provider
keyStore	KeyStore
aliases	ArrayList<String>
password	char[ ]
Métodos	
Nombre	SignerkeyStorePKCS11 (String arg0, char[ ] arg1)
Descripción	Constructor de la clase. Recibe en arg0 una cadena de configuración para inicializar el proveedor criptográfico, y en arg1 el pin de acceso al almacén de llaves.
Nombre	SignerkeyStorePKCS11 (InputStream arg0, char[ ] arg1)
Descripción	Constructor de la clase. Recibe en arg0 el fichero de configuración para inicializar el proveedor criptográfico, y en arg1 el pin de acceso al almacén de llaves.
Nombre	getPrivateKey(String arg0)
Descripción	Devuelve una referencia a la llave privada asociada a un alias. Recibe en arg0 el

	alias.
Nombre	getChain(String arg0)
Descripción	Devuelve la cadena de certificados asociada a un alias. Recibe en arg0 el alias.
Nombre	getAliases()
Descripción	Devuelve la lista de aliases del almacén de llaves.

Tabla 3.19 uci.pdfsigner.core.certificate.SignerkeyStorePKCS12

Paquete contenedor	uci.pdfsigner.core.certificate
Super clase	SignerkeyStore
Nombre	SignerkeyStorePKCS12
Descripción	
Clase encargada de abstraer el acceso al almacén de llaves, alojado en archivos compatibles con el estándar PKCS#12, que se utiliza en el proceso de firma digital. Implementa los métodos de la interfaz SignerkeyStore y utiliza el proveedor criptográfico org.bouncycastle.jce.provider.BouncyCastleProvider.	
Atributo	**Tipo**
provider	Provider
keyStore	KeyStore
password	char[ ]
Métodos	
Nombre	SignerkeyStorePKCS12 (InputStream arg0, char[ ] arg1)
Descripción	Constructor de la clase. Recibe en arg0 el almacén de llaves compatible con el estándar PKCS#12, y en arg1 la contraseña de acceso al almacén de llaves.
Nombre	getPrivateKey(String arg0)
Descripción	Devuelve la llave privada asociada a un alias. Recibe en arg0 el alias.
Nombre	getChain(String arg0)
Descripción	Devuelve la cadena de certificados asociada a un alias. Recibe en arg0 el

	alias.
Nombre	getAliases()
Descripción	Devuelve la lista de alias del almacén de llaves.

Tabla 3.20 uci.pdfsigner.core.certificate.CertUtil

Paquete contenedor	uci.pdfsigner.core.certificate
Super clase	java.lang.Object
Nombre	CertUtil
Descripción	
Clase que contiene métodos estáticos para el manejo de los atributos de certificados digitales.	
Métodos	
Nombre	getIssuerCN(String arg0)
Descripción	Devuelve la cadena que correspondiente a CN (common name) del atributo IssuerName del Certificado digital. Recibe en arg0 el valor de la cadena IssuerName.
Nombre	getIssuerO(String arg0)
Descripción	Devuelve la cadena correspondiente a O (organization) del atributo IssuerName del Certificado digital. Recibe en arg0 el valor de la cadena IssuerName.
Nombre	getSubjectCN(String arg0)
Descripción	Devuelve la cadena que correspondiente a CN (common name) del atributo SubjectName del Certificado digital. Recibe en arg0 el valor de la cadena SubjectName.
Nombre	getSubjectO(String arg0)
Descripción	Devuelve la cadena correspondiente a O (organization) del atributo SubjectName del Certificado digital. Recibe en arg0 el valor de la cadena SubjectName.
Nombre	getSubjectE(String arg0)
Descripción	Devuelve la cadena correspondiente a E (e-mail) del atributo SubjectName del Certificado digital. Recibe en arg0 el valor de la cadena SubjectName.
Nombre	getAtribute(String arg0, String arg1)
Descripción	Busca en la cadena srg0, el valor del atributo de la cadena arg1.

Tabla 3.21 uci.pdfsigner.core.timestamp.TSAClientBouncyCastle

Paquete contenedor	uci.pdfsigner.core.timestamp
Super clase	TSAClient
Nombre	TSAClientBouncyCastle
Descripción	
Clase proveedora de tokens de sellado de tiempo compatibles con el estándar RFC 3161.	
Atributo	**Tipo**
tsaURL	String
tsaUsername	String
tsaPassword	String
tokSzEstimate	int
useproxy	boolean
proxyUsername	proxyPassword
Métodos	
Nombre	TSAClientBouncyCastle(String arg0, String arg1, String arg2)
Descripción	Constructor de la clase. Recibe en arg0 la dirección URL del servidor de sellado de tiempo, en arg1 usuario del servidor, y en arg2 contraseña del servidor.
Nombre	getTimeStampToken(byte[] arg0)
Descripción	Devuelve un token de sellado de tiempo compatible con el estándar RFC 3161.
Nombre	getTSAResponse(byte[] arg0)
Descripción	Devuelve la respuesta del servidor de sellado de tiempo. Recibe en arg0 el resumen de los datos al que se le desea aplicar el sellado de tiempo.
Nombre	getTokenSizeEstimate()
Descripción	Devuelve el tamaño estimado del token.

3.6 Fase de Producción.

Con la culminación de las iteraciones comienza la fase de producción, donde se llevan a cabo ajustes del rendimiento del sistema con vista a obtener una aplicación más rápida, robusta y

lista para ser probada por el cliente. En esta fase se elaboran las pruebas de aceptación con el fin de lograr una retroalimentación del cliente donde generalmente se introducen nuevas funcionalidades o se rediseñan las existentes.

3.6.1 Pruebas de Aceptación

Las pruebas del cliente también conocidas como pruebas de aceptación son definidas por este y su objetivo es probar que las funcionalidades del sistema corresponden con las historias de usuarios definidas por él. Estas son pruebas de caja negra que se realizan a través de las interfaces del sistema, a continuación se muestran las pruebas de aceptación aplicadas al sistema:

Tabla 3.22 Prueba de aceptación Firma de Archivos PDF

Caso de Prueba de Aceptación	
Código: HU5_P1	**Historia de Usuario:** Firmar archivos PDF.
Nombre: Firma de archivos PDF.	
Descripción: Prueba de funcionalidad para la firma de archivos PDF.	
Condiciones de ejecución: Se debe contar con algún certificado digital para realizar la firma.	
Entrada / Pasos de Ejecución: **1.** El usuario especifica la ruta del archivo PDF o los archivos PDF a firmar. **2.** El usuario especifica la ruta de destino del archivo firmado. **3.** Selección de un certificado digital para la firma.	
Resultado esperado: El archivo es firmado correctamente, se muestra un mensaje informando al usuario del éxito de la operación.	
No conformidades:	
Evaluación de la prueba: Prueba satisfactoria.	

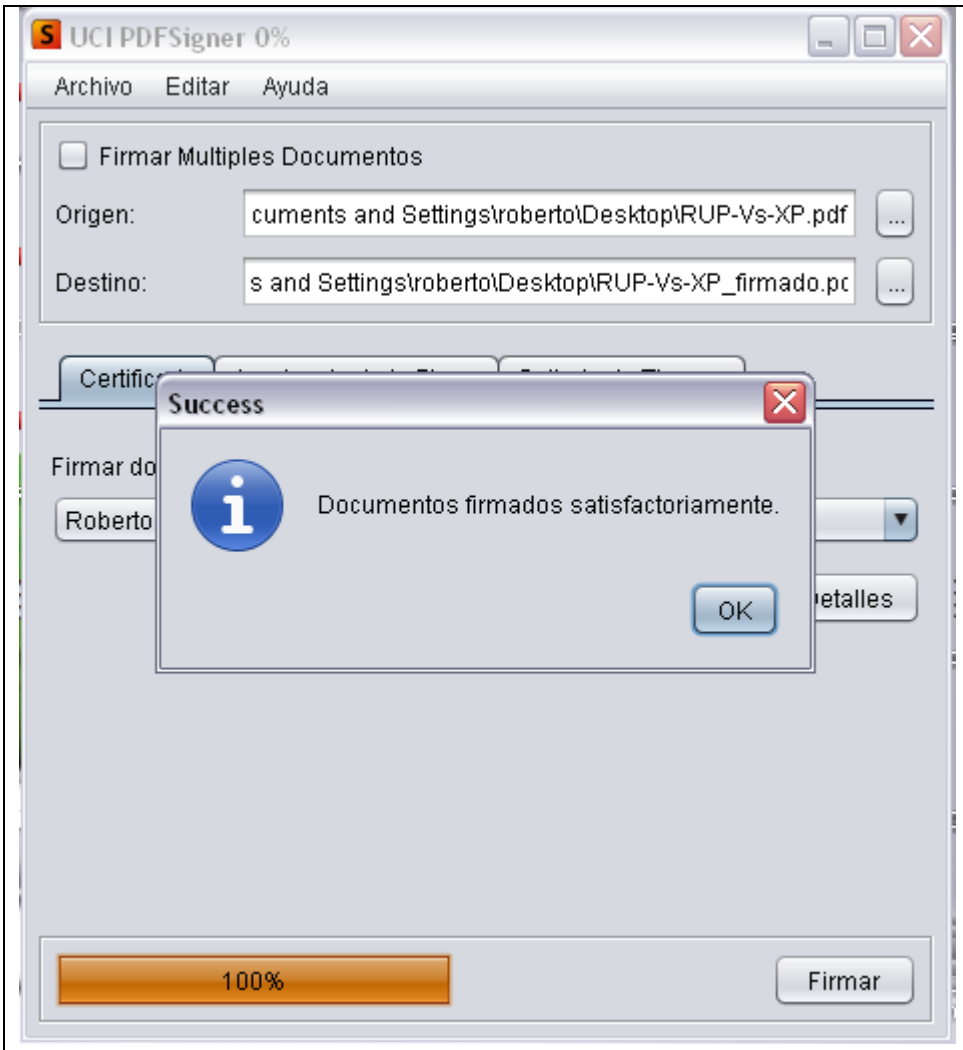

Tabla 3.23 Prueba de aceptación Gestión de Certificados Digitales.

Caso de Prueba de Aceptación	
Código: HU2_P1	**Historia de Usuario:** Gestión de Certificados Digitales
Nombre: Gestionar Certificados Digitales	
Descripción: Prueba de funcionalidad para la gestión de certificados digitales.	
Condiciones de ejecución:	
Entrada / Pasos de Ejecución: **1.** El usuario especifica la ruta del certificado digital que va a utilizar para firmar.	

2. El usuario introduce la contraseña del certificado seleccionado.
3. Si se cuenta con una tarjeta inteligente se carga el certificado digital que contiene.
4. Se introduce el pin.

Resultado esperado: Gestión satisfactoria de los certificados digitales, en caso de no introducir el pin o la contraseña el sistema debe mostrar un mensaje informándole al usuario del error.

No conformidades:

Evaluación de la prueba: Prueba satisfactoria

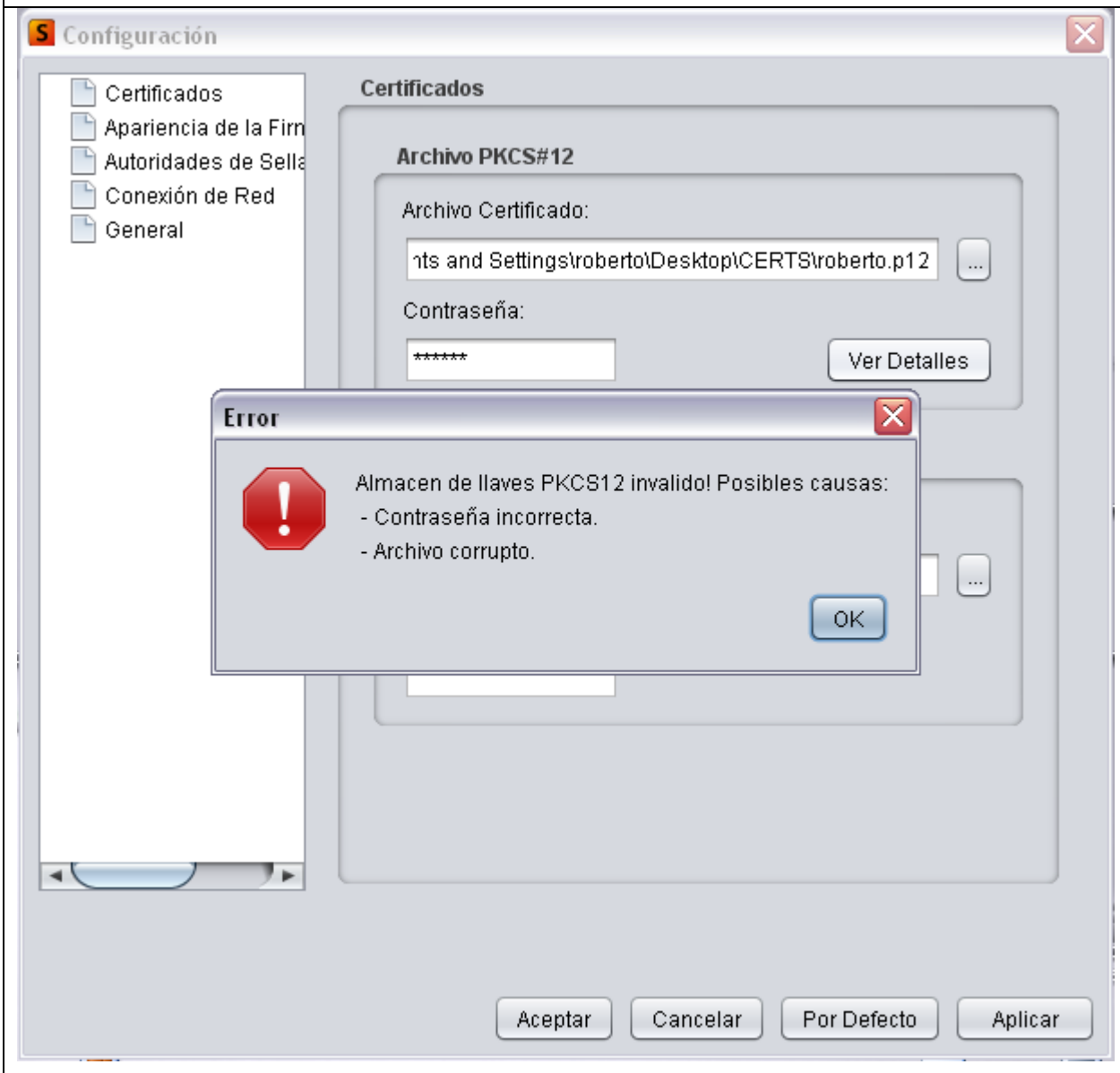

Tabla 3.24 Prueba de aceptación Gestión de Apariencia de la Firma Digital.

Caso de Prueba de Aceptación	
Código: HU3_P1	**Historia de Usuario:** Gestión de la Apariencia de la Firma Digital
Nombre: Gestionar Apariencia de la Firma Digital	
Descripción: Prueba de funcionalidad para la gestión de apariencia de la firma digital.	
Condiciones de ejecución:	
Entrada / Pasos de Ejecución: **1.** El usuario especifica la razón por la cual desea firmar. **2.** El usuario introduce el lugar desde donde se está realizando la firma. Ejemplo: Centro de Identificación y Seguridad Digital. **3.** Se especifican datos de la persona que está realizando la firma. **4.** Se selecciona una acción permitida para la certificación del documento. **5.** Para mostrar la firma el usuario puede escoger la posición y página donde desea que esta se muestre. **6.** Estos campos son opcionales, el usuario puede llenarlos o no.	
Resultado esperado: El archivo PDF debe contener un campo de firma visible con los datos	
No conformidades:	
Evaluación de la prueba: Prueba satisfactoria	

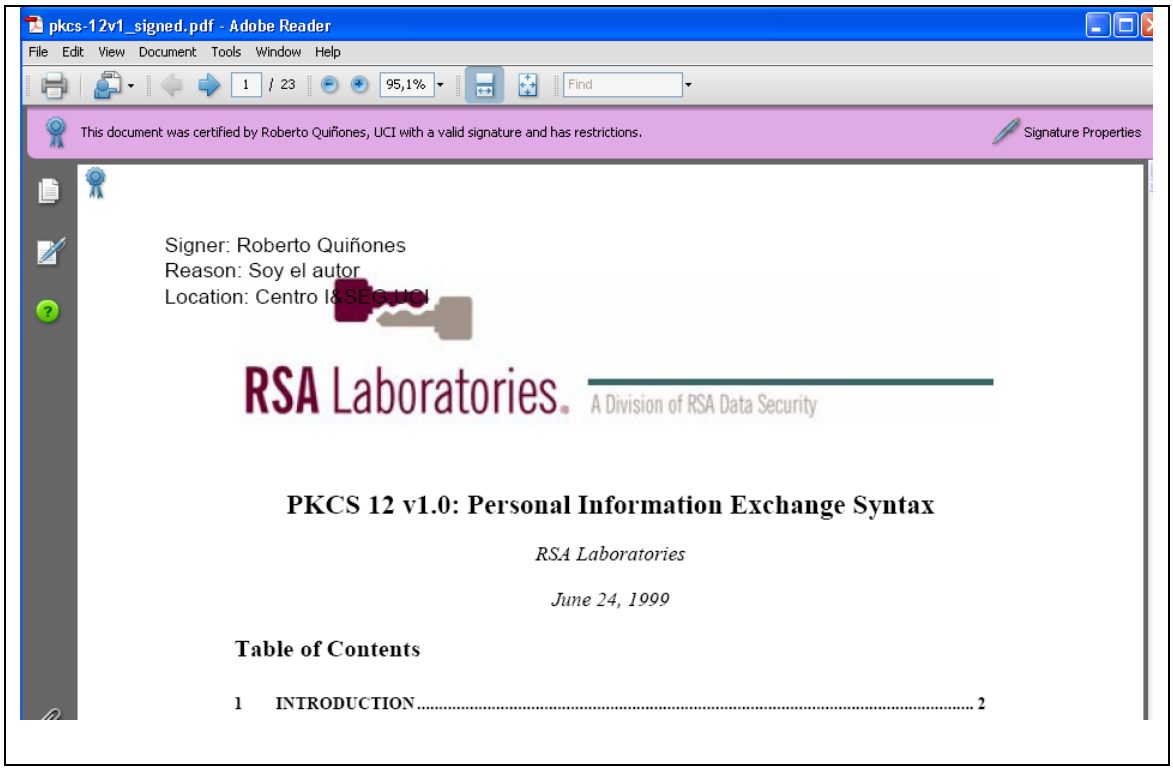

Tabla 3.25 Prueba de aceptación para la Gestión del Servicio de Sellado de Tiempo.

Caso de Prueba de Aceptación	
Código: HU4_P1	**Historia de Usuario:** Gestión del Servicio de Sellado de Tiempo
Nombre: Gestionar Servicio de Sellado de Tiempo	
Descripción: Prueba de funcionalidad para la gestión de sellado de tiempo.	
Condiciones de ejecución: Se debe tener habilitado la opción de sellado de tiempo.	
Entrada / Pasos de Ejecución: • El usuario selecciona un servidor de sellado de tiempo. • En caso de que el servidor requiera autenticación, el usuario debe introducir su contraseña.	
Resultado esperado: El documento PDF firmado con el sellado de tiempo	
No conformidades:	

Evaluación de la prueba: Prueba satisfactoria

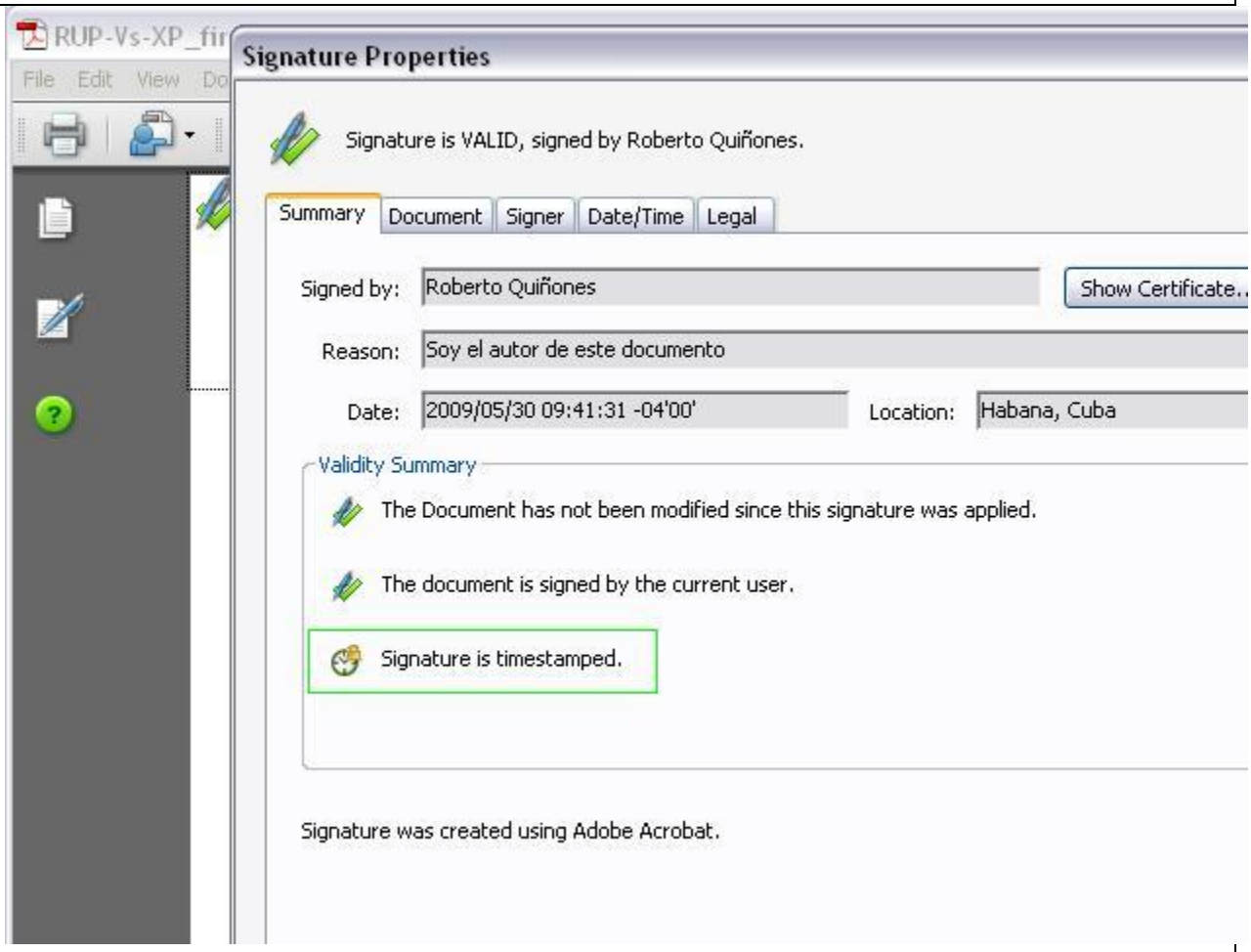

Tabla 3.26 Prueba de aceptación para la Gestión de la Configuración Guardada.

Caso de Prueba de Aceptación	
Código: HU6_P1	**Historia de Usuario:** Gestión de la Configuración
Nombre: Configuración Guardada.	
Descripción: Prueba de funcionalidad para el salvado de la configuración.	
Condiciones de ejecución:	

Entrada / Pasos de Ejecución: • El usuario selecciona la opción que desea configurar.
Resultado esperado: Se guardan los cambios realizados.
No conformidades:
Evaluación de la prueba: Prueba satisfactoria.

Tabla 3.27 Prueba de aceptación para la Gestión de la Configuración por Defecto.

Caso de Prueba de Aceptación	
Código: HU6_P2	**Historia de Usuario:** Gestión de la Configuración
Nombre: Cargar configuración por defecto.	
Descripción: Prueba de funcionalidad para la configuración por defecto.	
Condiciones de ejecución:	
Entrada / Pasos de Ejecución: • El usuario selecciona la opción que desea configurar.	
Resultado esperado: Cuando se inicie la aplicación sin los archivos de configuración, el sistema carga la configuración por defecto.	
No conformidades:	
Evaluación de la prueba: Prueba satisfactoria.	

3.7 Conclusiones del capítulo

Luego de terminadas las fases de Iteraciones a Primera Liberación y Producción de la solución propuesta, se concluye:

1. A partir del desglose de las historias de usuario en tareas de la ingeniería fue posible la realización del diseño y codificación.

2. El desarrollo guiado por pruebas aseguró la ejecución correcta de la solución en todo el período de desarrollo, aminorando el tiempo invertido en el ciclo compilación-ejecución.

3. Las pruebas de aceptación concluyeron de manera exitosa demostrando la satisfacción del cliente con la solución.

Conclusiones

Al término de esta investigación se concluye:

1. Se desarrolló una aplicación para la firma digital de documentos en formato PDF, que garantiza la autenticidad, integridad y no repudio de información almacenada en dichos documentos.

2. Se garantizó la validez a largo plazo de la firma digital a través del sellado de tiempo.

3. Con el uso de tecnologías Java se logró una aplicación multiplataforma, completamente funcional en los principales sistemas operativos existentes en el mercado.

Recomendaciones

Como producto de la culminación de la presente investigación se tiene en cuenta las siguientes recomendaciones:

1. Incluir verificación de documentos PDF firmados, haciendo uso de CRL y OCSP.
2. Explotar las características del formato PDF para garantizar la confidencialidad de la información a través del cifrado simétrico.
3. Incluir soporte para firmar digitalmente varios tipos de documentos electrónicos.
4. Añadir soporte a para múltiples algoritmos de firma digital como DSA y EDSA.
5. Desarrollar una versión para servidores.

Bibliografía

1. *Digital signature bill enables e-commerce.* **Jones, Jennifer and Johnston, Margret.** 25, 6 19, 2000, InfoWorld, Vol. 22.

2. **Valdés, Marisol.** BetSime - La Revista del Empresario Cubano. [Online] julio 2003. http://www.betsime.disaic.cu/secciones/fin_ja_03.htm.

3. **Lucena, Manuel.** *Criptografía y Seguridad en Computadores.* 2004.

4. **Burnett, Steve and Paine, Stephen.** *RSA Security's Official Guide to Cryptography.* s.l. : McGraw-Hill, 2001.

5. **Mogollon, Manuel.** *Cryptography and Security Services: Mechanisms and Aplications.* s.l. : Cybertech Publishing, 2007.

6. **Rankl, Wolfgang and Effing, Wolfgang.** *Smart Card Handbook.* 3ra. s.l. : John Wiley & Sons Ltd, 2002.

7. **Laboratories, RSA.** *RSA Laboratories' Frequently Asked Questions About Today's Cryptography, Version 4.1.* s.l. : RSA Security Inc., 2000.

8. **Adams, C., et al.** RFC:3161. *Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP).* 2001.

9. *Hardware Criptográfico en las PKI.* **Baeza, José T.** 9, 2007, www.revista-ays.com.

10. **Ceballos, Marlon.** Formato de Archivo PDF. [Online] http://issuu.com/xpert/docs/conferen_pdfs.

11. Adobe - Familia de Adobe Acrobat: formato de documento portátil de Adobe. [Online] www.adobe.com/es/products/acrobat/adobepdf.html.

12. **Solis, Camilo J. and Figueroa, Roberth G.** *Metodologías Tradicionales vs. Metodologías Ágiles.* s.l. : Universidad Técnica Particular de Loja, Escuela de Ciencias de la Computación.

13. teleformacion.uci.cu. [Online] 2008-2009. http://teleformacion.uci.cu/file.php/102/Curso_2008-2009/Materiales_Basicos/Materiales_Basicos_CTP_1/Material_Clase_Teorico_Practica_1.doc.

14. **Wells, J. Donovan.** Extreme Programming: A gentle introduction. [Online] 1999. http://www.extremeprogramming.org/.

15. **Beck, Kent.** *Extreme Programming Explained: Embrace Change.* s.l. : Addison-Wesley, 2000.

16. Java PKCS#11 Reference Guide. [Online] java.sun.com/javase/6/docs/technotes/guides/security/p11guide.html.

17. **Zubizarreta, Pérez Alexei.** Serie Científica. [Online] mayo 22, 2009. http://seriecientifica.uci.cu/articulos/firma-digital-avanzada-de-documentos-juridicos-en-actividades-registrales.

18. **Tom St Denis, Simon Johnson.** *Cryptography for Developers.* s.l. : Syngress Publishing, Inc., 2007.

19. **Navarro, Lourdes Pérez.** Informatización de notarias y registros venezolanos. *Diario Granma.* mayo 29, 2009, 149.

20. **Mollin, Richard A.** *RSA and Public-key Cryptography.* s.l. : CRC Press, 2003.

21. **Marchesi, Michele, et al.** *Extreme Programming Perspectives.* s.l. : Addison-Wesley, 2002. 0-201-77005-9.

22. *Signature Schemes and Applications to Cryptographic Protocol Design.* **Lysyanskava, Anna.** 2002.

23. **Jeffries, Ron, Anderson, Ann and Hendrickson, Chet.** *Extreme Programming Installed.* s.l. : Addison-Wesley, 2000. 0-201-70842-6.

24. **Adams, Carlisle and Lloyd, Steve.** *Understanding PKI: Concepts, Standards, and Deployment Considerations.* 2da. s.l. : Addison-Wesley, 2002.

25. **A. Menezes, P. van Oorschot, S. Vanstone.** *Handbook of Applied Cryptography.* s.l. : CRC Press, 1996.

26. *EU Approves a Law Giving Legal Status To Digital Signatures.* 107, 12 1, 1999, Wall Street Journal - Eastern Edition, Vol. 234.