Universidad de las Ciencias Informáticas

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Título:

Componente de cifrado de características biométricas utilizando plantillas cancelables

Autora:

Laura Elena González Varela

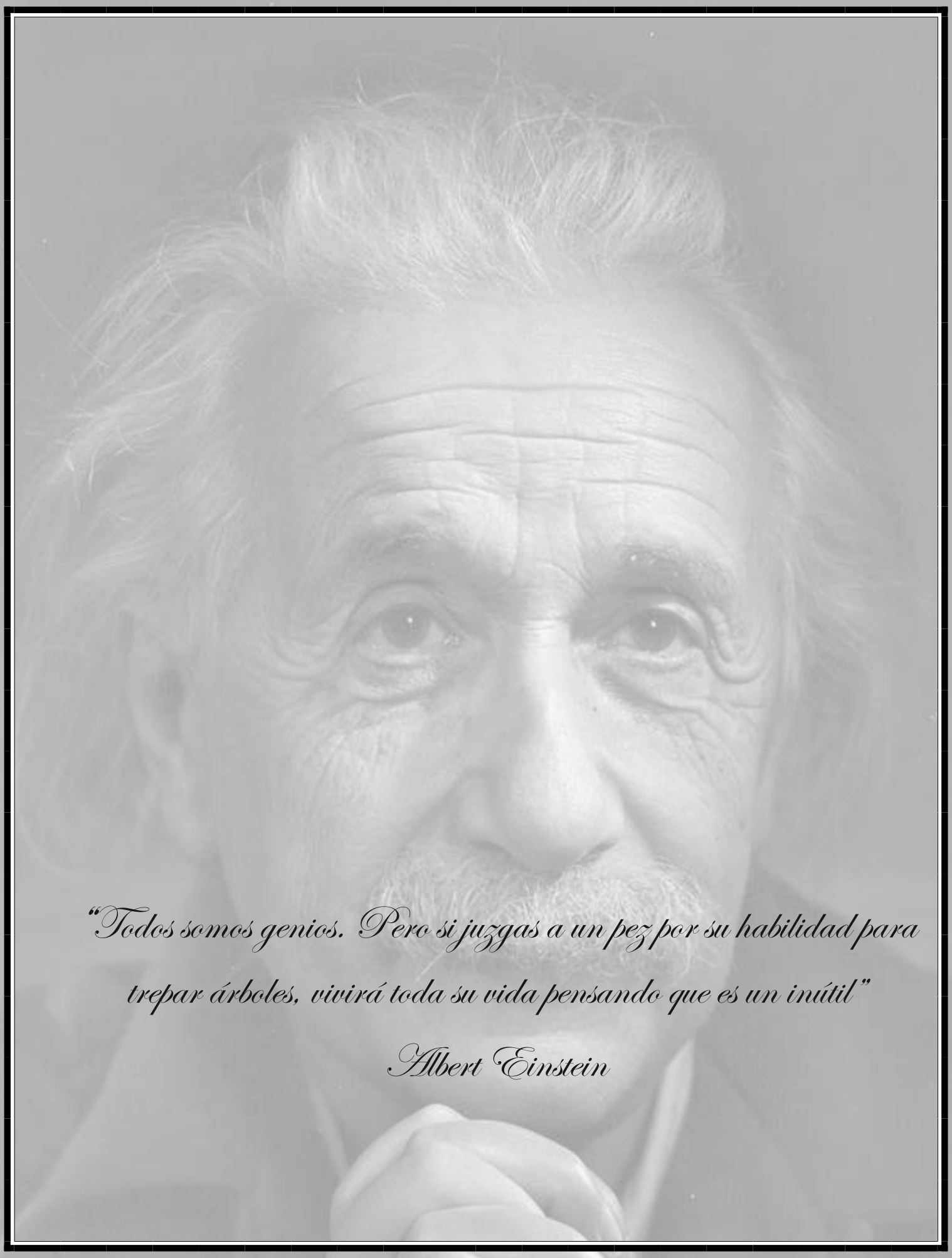
Tutores:

Ing. Dailín Martínez Pardo

Ing. Dashiel Hondarez Laza

Ing. José Carlos Pérez Zamora

La Habana, junio del 2017



"Todos somos genios. Pero si juzgas a un pez por su habilidad para trepar árboles, vivirá toda su vida pensando que es un inútil"

Albert Einstein

Declaración de autoría

Declaro por este medio que yo Laura Elena González Varela, con carné de identidad 94073036612 soy el autor principal del trabajo titulado "**Componente de cifrado de características biométricas utilizando plantillas cancelables**" y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

Laura Elena González Varela

Firma del Autor

_____ _____

Ing. Dailín Martínez Pardo Ing. Dashiel Hondarez Laza

Firma del Tutor Firma del Tutor

_____

Ing. José Carlos Pérez Zamora

Firma del Tutor

Datos del contacto

Tutor:

Ing. Dailín Martínez Pardo

Graduado de Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI) en el año 2015. Cuenta con 2 años de experiencia.

Universidad de las Ciencias Informáticas, La Habana, Cuba

Correo electrónico: dmpardo@uci.cu

Tutor:

Ing. Dashiel Hondarez Laza

Graduado de Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI) en el año 2015. Cuenta con 2 años de experiencia.

Universidad de las Ciencias Informáticas, La Habana, Cuba

Correo electrónico: dhondarez@uci.cu

Tutor:

Ing. José Carlos Pérez Zamora

Graduado de Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI) en el año 2013. Profesor Instructor de Bases de Datos del departamento docente de ingeniería y gestión de software. Actualmente se encuentra realizando su investigación de maestría. Cuenta con 4 años de experiencia.

Universidad de las Ciencias Informáticas, La Habana, Cuba

Correo electrónico: secarlos@uci.cu

Dedicatoria

A mi abuelo, que a pesar de no estar físicamente a mi lado, siempre me acompaña.

A mi mamá y abuela por ser mi apoyo y guía, por ser todo.

A mi hermano a quien amo sobre todas las cosas.

A mi papá por ser mi motor impulsor, por ser la persona que me motiva a seguir adelante por duro que el camino sea, por ser la persona por la cual me esfuerzo por ser mejor cada día, por no creer en mí,

A mi novio, por entrar en mi vida y cambiarlo todo.

Agradecimientos

Quisiera agradecer en primer lugar a mi persona favorita en el mundo, el caballero de mi cuento, mi héroe, la persona con quien tuve el enorme placer de compartir solo 8 años de mi vida, mi amado abuelo, siempre apoyándome entre las sombras con su inolvidable sonrisa.

A mi mamá, por ser la mujer maravilla, por mostrarme que siempre existe luz al final de camino, por ser mi mayor orgullo, por ser mi ejemplo a seguir, por haberme dado todo el amor del mundo.

A mi abuela, quien siempre tiene un consejo al alcance de mis oídos y un abrazo al alcance de mis pesares.

A mi hermanito, con quien me une el más fuerte de los lazos, a quien aunque se lo diga en muy pocas ocasiones amo de una forma casi inexplicable.

A Raisa y Alberto, por apoyarme y ayudarme siempre, por ser la familia que me regaló el destino, por quererme como lo hacen.

A Ovi, por ser la persona correcta en el momento correcto, por hacerme querer ser la princesa de tu cuento, por cambiar mi mundo de la forma en que lo hiciste, por amarme tanto.

A Baby, quien ha estado ahí para mí a lo largo de estos difíciles años, con quien reí tanto como lloré.

A Vismar, quien a pesar de la distancia aprendió a estar cerca.

A mis compañeros de aula, tanto los viejos como los nuevos, los que aún están como los que fueron rindiéndose en el camino, de todos ustedes me llevo un pedacito.

A los Hectors y a Heri, por siempre permitirme robarles sus conocimientos y muchas sonrisas, por todos los momentos que pasamos juntos.

A mis niñas lindas: Raisa, Lisbet, Wilbia y Betsy, gracias por llegar y cambiar mi pequeño mundo, por mostrarme que en muchas ocasiones las apariencias engañan, por los momentos especiales que vivimos juntas y por el apoyo que recibí de ustedes en los malos momentos.

A Arle, por todas las ocasiones en las cuales fuimos más que amigas hermanas, a pesar de la distancia que hoy nos separa te sigo queriendo mucho.

A mis profesores, por permitirme hoy ser una profesional, por nunca negarme su ayuda y apoyo.

A mis tutores, a Ramón y Dismey por todo el apoyo que me brindaron a lo largo del proceso de tesis.

A todos esos amigos o compañeros que hicieron posible este sueño, a los que me acompañaron por poco o mucho tiempo, a los que me sacaron una sonrisa en los momentos más difíciles.

Resumen

Con el aumento de las tecnologías, aumenta también la necesidad de garantizar la protección de la información. Los sistemas de autenticación biométricos son cada vez más utilizados, en ellos las huellas dactilares son los identificadores biométricos más empleados para el reconocimiento de personas. En un principio los puntos característicos extraídos de la huella se almacenaban en texto claro, existiendo gran posibilidad de alterar la información, tanto contenida en la base de datos, como la que viajaba por los diferentes módulos del sistema de identificación. Se decidió, para darle solución a este problema que una vez capturada la huella y extraídos estos puntos sean cifrados antes de ser almacenados, disminuyendo así la probabilidad de fuga de la información biométrica al ocurrir un ataque. Cuando el cifrado de las mismas se realiza por métodos tradicionales, al realizarse la verificación biométrica, la plantilla cifrada debe descifrarse con el objetivo de ser emparejada con las características de consulta, por lo cual los datos quedan expuestos durante cada intento de autenticación. La presente investigación tiene como finalidad desarrollar un componente que cifre los datos y efectúe la comparación directamente en el dominio cifrado, con el objetivo de mitigar la brecha de seguridad existente. Para lograrlo, se realizó un estudio del estado del arte de los diferentes modelos existentes. Se seleccionaron las herramientas necesarias para la realización del componente, el cual se diseña, implementa y realiza la validación para comprobar el correcto funcionamiento por medio de pruebas.

Palabras claves: ataques, cifrado, huellas dactilares, protección de la información, sistemas biométricos.

Índice de contenido

Índice de figuras

Índice de tablas

Introducción

En la actualidad, con el aumento de las tecnologías, crece la necesidad de garantizar la seguridad de la información y la intimidad personal. Los esquemas de autenticación actuales tales como las contraseñas, tarjetas de identificación, códigos de identificación personal, tarjetas inteligentes entre otros, pueden ser olvidados, sustraídos, perdidos o duplicados. La autenticación biométrica presenta baja probabilidad de ocurrencia de estas inevitables fugas de seguridad.

El término biometría se refiere al uso de características distintivas anatómicas o de comportamiento, denominadas rasgos o identificadores biométricos, para el reconocimiento automático de individuos (Maio, 2009). Entre los identificadores más empleados se encuentran: las huellas dactilares, la retina, el iris, los patrones faciales, la geometría de la palma de la mano. Uno de los métodos más populares para el reconocimiento biométrico es el basado en huellas dactilares, debido a que estas cumplen con las tres leyes fundamentales de la Dactiloscopia[1]: perennidad, inmutabilidad y diversidad infinita (Hernández León, 2011).

Según (Murillo-Escobar M.A, 2014) los procesos básicos en un sistema biométrico son: enrolamiento que consiste en extraer una o más muestras del identificador biométrico con un método (por ejemplo, extracción de minucias en huellas dactilares) y generar una plantilla para almacenarla en memoria (en sitio o remotamente) para futuras comparaciones; verificación que consiste en comparar los datos biométricos actuales con los almacenados (para autenticación uno a uno y para identificación uno a todos) para determinar si concuerdan.

El mayor riesgo dado en un sistema biométrico es la suplantación de la identidad del individuo mediante la imitación (la voz, la firma) o la reproducción (generación fraudulenta de la imagen dactilar o el iris) del rasgo a reconocer. Según (K. Jain, 2008) (Muñoz, 2010) las vulnerabilidades existentes, en un sistema de identificación biométrica, constituyen 8 puntos fundamentales de ataques: ataques al sensor o escáner, ataques al módulo extractor de características, ataques al módulo comparador, ataques a la base de datos del sistema, ataques a los distintos canales de comunicación entre los módulos, los cuales se pueden incluir en cuatro categorías: (i) los ataques en la interfaz de usuario, (ii) ataques en las interfaces entre los módulos, (iii) los ataques a los módulos, y (iv) los ataques a la base de datos donde están almacenadas las plantillas. Entre los ataques más comunes se encuentran los ataques de denegación de servicio y de intrusión, la

[1] **Dactiloscopia:** sistema de identificación mediante la comparación de las huellas digitales.

sobrescritura del extractor o el comparador de características, la intercepción del canal de comunicación entre estos, del canal de comunicación del comparador con la base de datos y la base de datos donde son almacenadas las plantillas.

Las plantillas de minucias, se transmiten y almacenan en texto claro, en el caso de los sistemas de identificación de huellas dactilares actuales. Debido a las vulnerabilidades existentes en este tipo de sistemas, se hace necesario para la seguridad de estos el uso de sistemas criptográficos. Para realizar la protección de los datos biométricos se han utilizado métodos de criptografía tradicional (K. Jain, 2012) con el objetivo de minimizar las fugas de seguridad de dichos datos. Estos métodos utilizan funciones para el cifrado que son extremadamente sensibles a variaciones, por lo cual una pequeña diferencia en los valores de los conjuntos de características extraídos de los datos biométricos conduciría a grandes cambios en las características cifradas resultantes. Los datos biométricos en su proceso de extracción, sufren variaciones de rotación, traslación, deformación no lineal y superposición parcial. Como consecuencia no se puede realizar la correspondencia biométrica directamente en el dominio cifrado, sino que la plantilla debe ser descifrada con el objetivo de ser emparejada con las características de consulta, por lo que los datos originales quedan expuestos durante cada intento de autenticación. El uso de algún virus o un mal funcionamiento de *hardware* posibilitan la obtención de la plantilla de minucias en texto claro durante este proceso.

Al ser obtenida la plantilla de minucias en texto claro (Cappelli, 2007) una de las fatídicas consecuencias es que la imagen de la huella dactilar sea reconstruida, para ser insertada en otros puntos del sistema biométrico como el módulo del sensor. Otra de las consecuencias más notorias es la pérdida del identificador biométrico de un individuo, dicha pérdida es para toda la vida, debido a que las huellas dactilares permanecen invariantes en el tiempo. De esta forma el atacante puede obtener acceso no autorizado en otros sistemas que utilicen el mismo identificador biométrico, constituyendo esta la **problemática** de la investigación.

Se define como **problema de la investigación:** ¿Cómo contribuir a la protección de los datos biométricos almacenados en las plantillas de minucias de huellas dactilares?
Como **objeto de estudio de la investigación:** El proceso de protección de plantillas de minucias de huellas dactilares.

Como **campo de acción:** Los componentes de protección de plantillas de minucias de huellas dactilares basados en plantillas cancelables.

Como **objetivo general de la investigación:** Desarrollar un componente para la protección de plantillas de minucias de huellas dactilares, que aumente la seguridad criptográfica de los datos biométricos con la utilización del modelo de plantillas cancelables.

Para lograr el objetivo general se han formulado las siguientes **preguntas de investigación**:

1. ¿Cuáles son los principales referentes teóricos que sustentan el uso de métodos de cifrado de características biométricas?
2. ¿Qué fundamentos presenta el análisis y diseño del componente de cifrado de características biométricas?
3. ¿Cuáles elementos debe contener un componente de cifrado de características biométricas utilizando plantillas cancelables, que contribuya a la protección de los datos biométricos almacenados en las plantillas de minucias de huellas dactilares?
4. ¿Cómo validar la contribución del componente de cifrado de características biométricas para la protección de las plantillas de minucias de huellas dactilares?

Para dar respuesta a las preguntas de investigación se trazaron las siguientes **tareas de investigación**:

1. Establecimiento de los principales referentes teóricos que sustentan el uso de métodos de cifrado de características biométricas.
2. Fundamentación de las metodologías de desarrollo, tecnologías y herramientas a utilizar en el desarrollo del componente.
3. Diseño de un componente de cifrado de características biométricas utilizando plantillas cancelables, que contribuya a la protección de los datos biométricos almacenados en las plantillas de minucias de huellas dactilares.
4. Realización de las pruebas definidas para validar el componente de cifrado de características biométricas utilizando plantillas cancelables.

Métodos científicos:

- **Analítico-sintético:** se utiliza para examinar y estudiar las investigaciones que se han desarrollado, descomponiéndolas, analizándolas y extrayendo de ellas la información necesaria para la investigación en curso, permitiendo adquirir conocimientos y consolidar los ya existentes.
- **Histórico-lógico:** facilita el análisis, de forma cronológica, de los métodos criptográficos de seguridad de plantillas de minucias de huellas dactilares resaltando sus características más notorias.

Métodos empíricos:

- **Observación:** es utilizado en el diagnóstico del problema en cuestión y permite un mejor diseño de la investigación. Posibilita estudiar los procesos de forma general, sin adentrarnos en sus especificaciones.
- **Experimental**: posibilita comprobar las funcionalidades en la medida que estas sean implementadas, con el objetivo de validar si cumplen o no con los requerimientos establecidos.

El trabajo estará estructurado de la siguiente forma:

Capítulo 1. Fundamentación teórica: se realiza un estudio del estado del arte de métodos de cifrado de características biométricas existentes, se abordan elementos teóricos vinculados a la investigación. Se analizan y seleccionan metodologías de desarrollo, tecnologías y herramientas a utilizar en el desarrollo del componente.

Capítulo 2. Propuesta de solución: se describe la solución propuesta y se ofrecen detalles de los principales aspectos relacionados con esta. Se efectúa el levantamiento de los requisitos funcionales y no funcionales y se determina la arquitectura a utilizar prestando una especial atención a los patrones de diseño empleados.

Capítulo 3. Implementación y pruebas: se describen los aspectos relacionados con la implementación, los estándares y las pruebas realizadas al componente, para validar el correcto funcionamiento de los requerimientos especificados.

Finalmente se presentan las Conclusiones y Recomendaciones derivadas de la investigación, las Referencias Bibliográficas, así como los Anexos que apoyan la comprensión y dan información adicional sobre el trabajo realizado.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo se realiza un análisis de los principales conceptos para facilitar un mejor entendimiento del problema planteado, así como el estudio del estado del arte de métodos de cifrado de características biométricas. Se seleccionan además la metodología, tecnologías y herramientas de desarrollo a utilizar durante el proceso de implementación de la solución.

1.1 Conceptos asociados

Biometría: se refiere al uso de características distintivas anatómicas o de comportamiento, denominadas rasgos o identificadores biométricos, para el reconocimiento automático de individuos (Maio, 2009). Tales como rostro, iris, huella dactilar, geometría de mano, firma, voz, entre otros (características universales, inalterables, únicas y medibles de cada persona).

Figura 1. "Rasgos físicos y conductuales utilizados en biometría"

Sistema biométrico: se trata básicamente de un sistema de reconocimiento de patrones que funciona del siguiente modo: el sensor captura un rasgo biométrico; se extraen un conjunto de características tras procesar la señal (patrón); y se comparan con las características de los usuarios (plantillas o *templates*) que hay almacenadas en la base de datos. En función de los resultados obtenidos por el comparador y el umbral de decisión, el sistema caracteriza el patrón de entrada como válido o no (Muñoz, 2010).

Reconocimiento biométrico: es un modo de referirse indistintamente al proceso de verificación o identificación de personas, responde a un sistema automático (Duró, 2009) basado en la inteligencia artificial y el reconocimiento de patrones, que permite la identificación y/o verificación de la identidad de personas a partir de características morfológicas o de comportamiento, propias y únicas del individuo.

Huella dactilar: una huella dactilar es la impresión visible o moldeada que produce el contacto de las crestas papilares de un dedo de la mano (generalmente se usan el dedo pulgar o el dedo índice) sobre una superficie (Muñoz, 2010). Estas tienen propiedades tales como: perennes, invariantes hasta la descomposición y con capacidad regenerativa diversiformes, no existen dos impresiones idénticas producidas por dedos diferentes.

Minucias: una minucia es un punto de interés de la huella digital. Las minucias tienen la siguiente representación: $\{\boldsymbol{x}, \boldsymbol{y}, \theta, \boldsymbol{t}\}$ donde $\boldsymbol{x}$, $\boldsymbol{y}$ es la posición en la imagen de la huella y $\boldsymbol{\theta}$ es el ángulo (en radianes) de dicha minucia. Las minucias se pueden dividir en varios tipos (figura 2).

Figura 2. "Tipos de minucias"

Seguridad en criptosistemas biométricos:

Según una compilación de (K. Jain, 2008) el cifrado de los datos que circulan por la red permiten al sistema protegerse de ataques producidos por la lectura/escritura de los paquetes de datos que circulan por la red de comunicación del sistema. El objetivo principal del cifrado es evitar que cualquier posible atacante pueda analizar la información que circula por el canal de comunicación. La criptografía de características biométricas permite los tres principios básicos de la seguridad: confidencialidad, privacidad e integridad, lo cual contribuye a erradicar la posibilidad de pérdida de los datos de esta índole.

Seguridad en las plantillas biométricas:

Un esquema de protección de plantilla biométrica ideal debe poseer las siguientes cuatro propiedades (K. Jain, 2013):

- Seguridad criptográfica: consiste en la dificultad computacional de obtener el conjunto de datos biométricos originales a partir del conjunto de datos biométricos protegidos.
- Revocabilidad: consiste en la posibilidad de obtener múltiples plantillas biométricas seguras a partir de los mismos datos biométricos originales.
- Rendimiento biométrico: Consiste en la capacidad del modelo de mantener el rendimiento del proceso de reconocimiento en términos de tasas de falsos aceptados y tasas de falsos rechazos.

1.2 Modelos para la protección de plantillas de huellas dactilares

Los modelos para la protección de plantillas de huellas dactilares según (K. Jain,2012) se clasifican en dos grandes grupos: basados en minucias y basados en patrones. A su vez los basados en minucias se pueden dividir en: los que aseguran directamente el conjunto de la representación no ordenada de las minucias, los que aseguran un nuevo conjunto de características desordenadas derivadas de las minucias y los que proveen un vector característico de longitud fija derivado de las minucias. Los modelos basados en patrones derivan directamente en un vector característico de longitud fija basado en la textura global del patrón de la huella dactilar.

Generalmente los modelos basados en minucias son los utilizados para la protección de plantillas de huellas dactilares, por lo cual se decide utilizar un modelo basado en minucias (Maio, 2009)**.**

1.2.1 Biohashing

El modelo de protección de *biohashing* (K. Jain, 2009) es intrínsecamente un acercamiento de dos factores donde la plantilla de la huella digital es transformada usando una función cuyos parámetros están definidos por una llave externa. En *biohashing* las plantillas protegidas no son reversibles a menos que esta y la llave sean conocidas simultáneamente.

Según (K. Jain, 2009) este modelo de protección se divide en dos componentes: la transformación integral invariante y discriminativa de los datos de una huella dactilar, con un moderado grado de tolerancia de

desvío y la discretización de los datos mediante un producto interno de números aleatorios (*tokens*) y datos del usuario.

Es exclusivamente aplicado a la característica de la textura de una huella digital extraída usando la técnica *FingerCode* **(Figura 3).** El centro de este método es cada minucia, permitiendo la protección de cada una y concatenando el resultado en una cadena. El término conocido como *MinuCode*, es el proceso *FingerCode* aplicado a cada minucia. (Fernández, 2016)

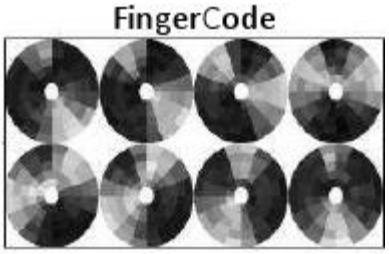


Figura 3. "Representación de un FingerCode"

El proceso relacionado (Belguechi, 2010) se describe como:

1. Extraer la plantilla de minucias de la imagen sin ser procesada previamente.
2. Calcular para cada minucia su *fingercode*.
3. Calcular el *minucode* de cada *fingercode*.
4. Realizar el proceso de *biohashing* a cada *minucode*.

Proceso de *biohashing*:

Ya extraída la plantilla de minucias de huellas dactilares, para cada minucia m se valida la región de interés que la rodea; esta región está determinada por divisiones circulares y estas a su vez se dividen en sectores S utilizando bandas B de anchura b. Cada banda está dividida en 16 sectores de igual ángulo (22,5º). Esta región es válida si está en el límite de la imagen y cada sector representa una alternación de crestas y valles. Esta alternancia se expresa a partir de la energía E del Espectro de Fourier[2]; si E>Tr (Tr es un umbral óptimo Otsu[3]) entonces S es válida. Posteriormente se filtra la región en ocho direcciones diferentes usando

[2] **Espectro de Fourier:** espectro de frecuencias de una función.
[3] **Umbral óptimo Otsu:** método que permite calcular el mejor valor umbral automáticamente, utilizando la varianza como medida de dispersión de los niveles de gris.

un banco de filtros de Gabor[4]. Contrariamente al método original, no es necesario normalizar la región puesto que se trabaja con una imagen binaria sin contraste.

Sea $Im_{i\theta}$ para θ (θ es la dirección de la imagen filtrada por el sector S_i, donde $i_{(1..B*16)}$). El vector característico o el *minucode* serán:

$F = (f_1, \dots, f_u)$ donde $u = B * C * D$ siendo B la cantidad de bandas, C el número de sectores en los que se divide cada banda, en este caso 16 y D las direcciones en las que se filtró la región, 8 en este caso, con $\forall f_{i\theta} \in$ F, $f_{i\theta} = \frac{1}{n_i} \sum_{n_i} | Im_{i\theta}(x,y) - p_{i\theta}|$, donde n_i es el número de píxeles en S_i y $p_{i\theta}$ es su media.

Para cada *minucode* se realiza el proceso de creación del *biohashing* de la siguiente forma:

1. Generar un conjunto de vectores aleatorios Γ.
2. Aplicar el proceso de Gram-Schmidt[5] para transformar el Γ base en un conjunto ortogonal de matrices $r_{\perp i}$, donde i va desde 1 hasta v, siendo v menor o igual que u.
3. Calcular el producto interno entre la función biométrica f y $r_{\perp i}$ ($f \mid r_{\perp i}$), donde $i_{(1..v)}$.
4. Calcular una cadena de bits de *biohashing* denotado b (b ∈ $\{0,1\}^v$).

$b_i = \begin{cases} 0 \; si \; (f \mid r_{\perp i}) \leq t \\ 1 \; si \; (f \mid r_{\perp i}) < t \end{cases}$, donde t es el umbral preestablecido.

La cadena de bits resultante b, llamada *biocode* representa la función de cada minucia. Solo se almacenarán *biocodes* para realizar la comparación.

La comparación de dos vectores se realiza (según Belguechi, 2010) mediante:

1. Correcta rotación.
2. Proceso de *biohashing* al conjunto de *minucodes* extraídos de la toma.
3. Llevar a cabo el algoritmo de coincidencia local entre las dos plantillas correlacionadas.

[4] **Banco de filtros de Gabor:** funciones que pueden diseñarse como un banco de filtros con diferentes dilataciones y rotaciones.
[5] **Gram-Schmidt:** Es un algoritmo que permite, a partir de un conjunto de vectores linealmente independientes de un espacio prehilbertiano, construir otro conjunto ortonormal de vectores que genere el mismo subespacio vectorial.

Sea $T = \{t1, \ldots, tm\}$ y $P = \{p1, \ldots, pn\}$ para las listas de *biocodes* extraídas de la plantilla y de la entrada de huellas dactilares respectivamente. El objetivo es encontrar un punto pj en P que corresponda exclusivamente a cada punto ti en T si este existe.

Debido a la combinación de factores como la variación intra-usuario[6] o la distorsión no lineal, un algoritmo fiable de comparación de minucias sigue siendo un problema difícil. En términos generales, los algoritmos pueden ser clasificados en coincidente global, coincidente local o una combinación de estos. Cuando se compara con enfoque global, los algoritmos de comparación de tipo coincidentes locales son más robustos a las distorsiones no lineales y superposiciones parciales.

Algoritmo de comparación de tipo coincidente local:

Primeramente se define una vecindad local de minucias m_i $\{m_1, \ldots, m_k\}$ de k vecinos más cercanos de m_i en términos de Distancia Euclidiana[7].

El algoritmo consta de dos fases:

1. Consiste en la selección del mejor par coincidente (root1, root2) / root1 ∈ T y root2 ∈ P utilizando la técnica de estimación de costos propuesta en (Belguechi, 2010).
2. Considerando a root1 y root2 como primeros nodos a explorar en las listas de *biocodes* extraídas (T, P), se debe hacer coincidir los k vecinos de root1 con los k vecinos de root2. Para hacer coincidir dos vecinos se utiliza una técnica de programación dinámica con una función de coste igual a la Distancia de Hamming[8] entre *biocodes.* Finalmente se aplica la función de consolidación propuesta en (Belguechi, 2010):

   $P = \frac{\text{nb par igualado}}{\text{mínimo (m,n)}}$, donde m es el tamaño de T y n el tamaño de P.

Desventajas (K. Jain, 2008):

[6] **Variación intra-usuario:** En un mismo usuario, dos impresiones distintas de un dedo tienen diferente número y ubicación de las minucias debido a la traslación y rotación que se manifiesta en una impresión con respecto a la otra.
[7] **Distancia Euclidiana:** distancia entre dos puntos, la cual se deduce a partir del teorema de Pitágoras.
[8] **Distancia de Hamming:** número de bits que tienen que ser cambiados para transformar una secuencia de código válida en otra y que esta continúe siendo válida.

- Si la llave especificada por el usuario es comprometida, la plantilla es ya no segura, porque la transformación es usualmente invertible, es decir, si un adversario obtiene acceso a la llave y la plantilla transformada, puede reconstruir la plantilla biométrica original.
- Mediante un ataque es posible que el atacante pueda obtener el *biocode*.
- Necesita ser diseñado de tal manera que la función de reconocimiento no se degrade, especialmente en presencia de variaciones grandes intra-usuario.

1.2.2 Bóveda difusa

Esencialmente implica codificar una plantilla como los coeficientes de un polinomio, y guardarlos como un conjunto A de elementos. Los elementos de A, junto con su proyección en el polinomio, forman el conjunto de la llave. Un número de pares falsos se mezclan también con el conjunto verdadero de la llave para formar la bóveda. Durante la verificación, un conjunto B de elementos se replantea. Cada elemento de B determinado es comparado con todos los pares en la bóveda para encontrar un elemento que hace juego (Arakala, 2006).

Descripción del proceso:

Se toma el conjunto (Arakala, 2006) SE = {x1, x2,-, xr} para denotar una plantilla biométrica que consiste en un conjunto de r puntos de un campo finito F. Con el objetivo de asegurar SE, se genera una clave criptográfica Kc uniformemente aleatoria de bits de longitud L y esta clave se transforma en un polinomio P de grado k (k < r) sobre F. Todos los elementos de la SE son luego evaluados en este polinomio para obtener el conjunto $\{P(xi)\}_{i=1}^{r}$. El conjunto de puntos $\{(xi, P(xi))\}_{i=1}^{r}$ se asegura entonces al ocultarlo entre un gran conjunto de puntos basura[9] q generados aleatoriamente $\{(aj, bj)\}_{j=1}^{q}$ que no se encuentran en el polinomio P (i.e., bj ≠ P(aj) y aj ∉ SE, $\forall j_{\{1,q\}}$. El conjunto de puntos auténticos y basuras, junto con sus evaluaciones polinómicas constituyen el dibujo yc o bóveda. Durante la autenticación, si al establecer la consulta biométrica SA está suficientemente cerca de la SE, el polinomio P puede ser reconstruido con éxito mediante la identificación de los puntos auténticos en yc que están asociados con SE.

[9]**Puntos basura:** puntos aleatorios cuya función es enmascarar los puntos auténticos que representan la información de la huella.

Existen parámetros fundamentales en el modelo de bóveda difusa que son r, q y k, r denota el número de puntos en la bóveda que se encuentran en el polinomio P y que dependen del número de características que se extraen de la plantilla. El parámetro q representa el número de puntos basura que se agregan, este parámetro influye en la seguridad de la bóveda, pues si no se añaden estos puntos la bóveda revela la información sobre la plantilla y el secreto. A medida que se añaden más puntos de este tipo incrementa la seguridad. Típicamente, el número de puntos basura es un orden de magnitud mayor que el número de puntos auténticos ($q >> r$). El parámetro k denota el grado del polinomio de codificación y controla la tolerancia del sistema a los errores en los datos biométricos.

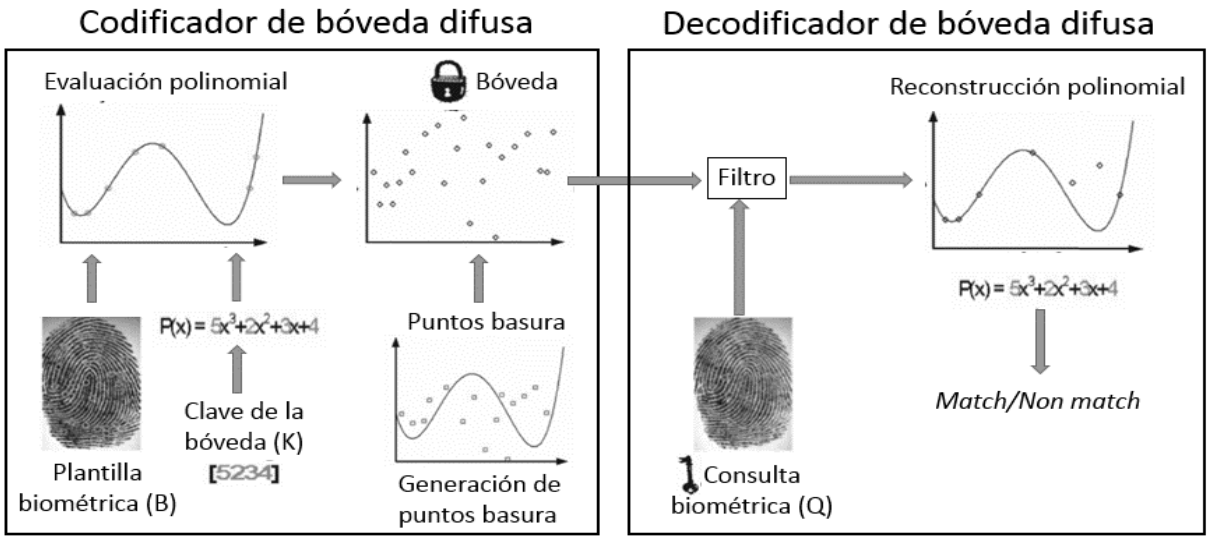


Figura 4. "Esquema de protección de plantillas de minucias usando bóveda difusa"

En el modelo (Arakala, 2006) más actual se plantea una modificación con respecto a la bóveda difusa clásica, la cual tiene como ventajas una decodificación única, un tamaño menor de la bóveda para la misma entropía y la cual necesita menor cantidad de puntos basura debido a los múltiples conjuntos que se insertan en la bóveda. Este enfoque permite un esquema funcional y un menor espacio en memoria, pero posee limitaciones en cuanto al tamaño de la lista resultante, el cual puede ser muy grande, por lo que existen restricciones en el número de conjuntos en el esquema y el espacio de la memoria aumenta rápidamente.

1.2.3 Plantillas cancelables

Las plantillas cancelables o no invertibles (Ratha, 2006) en lugar de almacenar la plantilla de minucias original, las características, la posición de minucias y orientaciones son transformadas de forma irreversible. Las plantillas cancelables plantean tres distintos tipos de transformaciones: transformaciones cartesianas, transformaciones polares y transformaciones funcionales.

La transformación Cartesiana (según Ratha, 2006) las posiciones de minucias están medidas en coordenadas rectangulares con referencia al corazón alineando el eje de las abscisas con su orientación. Este sistema de coordenadas está dividido en celdas de tamaño fijo. La transformación consta de cambiar las posiciones de la celda, donde cada celda (que puede contener una o más minucias) se desplaza a una nueva posición en la cuadrícula correspondiente a las traducciones establecidas por la clave generada aleatoriamente.

La transformación polar (Fernández.R.S**,** 2016): las posiciones de minucias son ahora medidas en coordenadas polares con referencia a la posición de fondo y los ángulos están medidos con referencia a la orientación. La transformación consta de cambiar las posiciones del sector. Los ángulos de minucias también cambian en conformidad a la diferencia en las posiciones del sector antes y después transformación. A diferencia de la transformación cartesiana, el mapeo espontáneo no es factible, desde que el d0 del ángulo es transformado para rd0 en una r de distancia del centro. Esto conduce situaciones donde puede ocurrir que los pares de minucias salten la distancia el uno del otro antes del ninguna de transformación, porque la distancia radial del sector transformado es muy diferente de la distancia radial de la posición original. Por consiguiente, en la transformación radial, el mapeo es gobernado por una llave restringida de traducción.

Transformación funcional: el requisito mínimo de traducción plantea una restricción severa en la elección de función (Ratha, 2006). Esta transformación se realiza utilizando una mezcla de funciones gaussianas bidimensionales (2D) y el campo de potencial eléctrico en una distribución de carga al azar 2D como un medio para traducir las minucias. Para lograr la transformación se evalúan las minucias en la función propuesta en (Ratha, 2007). Cada minucia transformada se representa con las medidas magnitud y gradiente. La primera denota la dimensión de la traducción correspondiente a la minucia y la segunda refleja la dirección de traslación de la minucia.

En la siguiente figura a, b y c representan las transformaciones (cartesiana, polar y funcional en este orden) a minucias originales.

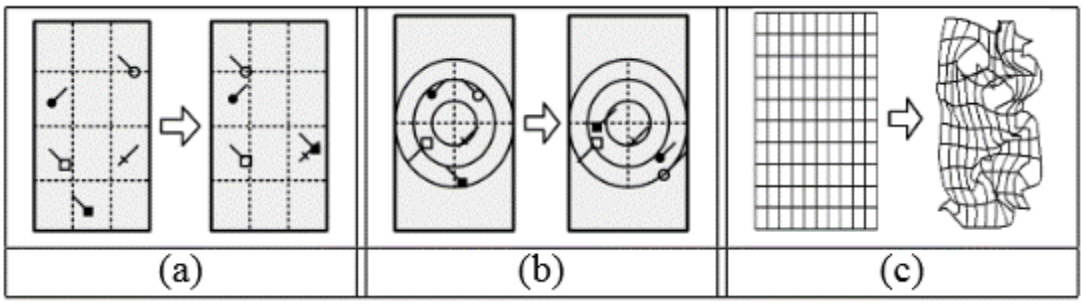


Figura 5. "Esquema de transformaciones para la protección de plantillas cancelables"

1.3 Selección del modelo a utilizar

En el análisis anterior se abordaron aspectos importantes de los distintos modelos de protección de plantillas de minucias de huellas dactilares encontrados, mediante el cual se ha decidido utilizar el modelo de plantillas cancelables por las siguientes razones:

- En el modelo de bóveda difusa si la B determinada es sustancialmente traslapada en la A determinada, los puntos basura pueden ser identificados y la p polinómica de la construcción es obtenida por el proceso de reconstitución usando el código de corrección de errores.
- En el modelo *biohashing* si la llave especificada en un usuario es comprometida, la plantilla es ya no segura, porque la transformación es usualmente invertible, es decir, si un adversario gana acceso a la llave y la plantilla transformada, puede obtener la plantilla biométrica original. Mediante un ataque es posible que el atacante pueda obtener el *biocode*. Necesita ser diseñado de tal manera que la función de reconocimiento no se degrade, especialmente en presencia de grandes variaciones intra-usuario.
- En el modelo basado en plantillas cancelables tanto en el espacio original como en el transformado las características del mismo usuario tienen alta similitud y las de diferentes usuarios son muy diferentes después de la transformación. La transformación debe ser no invertible, pues a partir de un conjunto de características transformadas resulta difícil para un adversario obtener el conjunto de características originales (o al menos una aproximación de la misma).

1.4 Alineación de plantillas de minucias de huellas dactilares

Un algoritmo de alineación (Pró, 2012) realiza cálculos mediante parámetros de rotación y traslación que conducen al mayor nivel de correspondencia espacial al ajustar la huella parametrizada con el patrón. Según (K. Jain, 2012) pueden ocurrir varias distorsiones durante la adquisición de datos biométricos: rotación, traslación, deformación no lineal y superposición parcial.

Entre los modelos de alineación existentes se pueden encontrar (Arathi, 2007):

- **Modelo de estructuras de minucias:** son los rasgos que caracterizan a la información relativa entre dos o más minucias. Estas estructuras son invariantes a la rotación y traslación debido a que tales características se obtienen con respecto a la ubicación y orientación de las minucias; significando esto una de sus principales ventajas. Un enfoque alternativo es extraer y almacenar algunos puntos de referencia a partir de la imagen de la huella capturada. Durante la autenticación, los puntos de referencia también pueden obtenerse a partir de la imagen de la huella de consulta. Los conjuntos de minucias de la plantilla almacenada y la de consulta pueden alinearse sobre la base de los parámetros obtenidos mediante la alineación de los correspondientes puntos de referencia (Jeffers, 2007).
- **Modelo de determinación de los puntos singulares**: existen diferentes enfoques como el método del índice de Poincaré, el método geométrico, el método de filtro complejo, entre otros, para obtener los puntos singulares en una imagen de huella dactilar. La exactitud de estos métodos está limitada por la baja calidad de la imagen de la huella capturada, la carencia de los puntos singulares en la clase arco y la naturaleza parcial de muchas imágenes de huellas dactilares capturadas utilizando sensores, pues existe la posibilidad de la no aparición de los puntos singulares.
- **Modelo punto focal de localización**: para la detección del punto de referencia a emplear en la alineación. El punto focal se define como el centro promedio de curvatura de una huella dactilar y constituye el centroide de todos los puntos de cruce, siendo un punto de cruce un punto de intersección de dos líneas normales de las crestas. Entre las limitaciones que presenta este modelo se encuentran su naturaleza iterativa (de ahí el alto requisito computacional) y la necesidad de seleccionar cuidadosamente el punto focal para la primera iteración.
- **Modelo de minucia estable**: la alineación basada en un punto de referencia de este tipo es sencillo y computacionalmente eficiente, sin embargo es difícil determinar el punto característico estable de forma

fiable. Un pequeño error en la localización del punto de referencia podría dar lugar a una alta tasa de falso rechazo.

- **Modelo de determinación del punto más cercano**: es un modelo utilizado para alinear la plantilla almacenada y la imagen de consulta sobre la base de la extracción de un conjunto de puntos con alta curvatura desde el campo de orientación de la huella dactilar. Un conjunto de puntos con alta curvatura es un conjunto de segmentos lineales por partes, cuya dirección de la tangente en cada punto es paralela a la dirección del campo de orientación en ese punto. Esta técnica de alineación no es computacionalmente eficiente; al almacenar muchos puntos de alta curvatura puede filtrarse más información acerca del patrón de la huella dactilar.

1.4.1 Selección del modelo de alineación a utilizar

De los modelos de alineación antes expuestos, para la realización del trabajo se decide utilizar el modelo de estructuras de minucias. Este trabaja con una serie de características que no varían mediante la rotación, traslación, además el grado de deformación lineal disminuye. Los modelos basados en puntos: puntos singulares, el punto focal de localización y el punto más cercano son modelos altamente sensibles a la obtención de las minucias. Es necesario para una primera iteración seleccionar cuidadosamente el punto focal de localización, debido a su naturaleza iterativa. Es difícil determinar el punto característico estable de forma fiable.

1.5 Metodología de desarrollo

Una metodología de desarrollo de *software* es un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a trazar las bases para un nuevo sistema. Las metodologías de *software* pueden ser clasificadas en dos grandes grupos. Las denominadas metodologías tradicionales, enfocadas en el control del proceso, con un riguroso seguimiento de las actividades involucradas en ellas y las metodologías ágiles, enfocadas en el factor humano, en la colaboración y participación del cliente en el proceso de desarrollo (RG Figueroa, 2008) (OT Gómez, 2014).

Metodologías Ágiles

Comparativa ágil vs tradicional

Metodologías agiles	Metodologías tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos

Figura 6. "Comparación entre metodologías ágiles y tradicionales"

1.5.1 Metodologías tradicionales

Las metodologías tradicionales proponen una disciplina de trabajo sobre el proceso de desarrollo de *software*, con el fin de conseguir un *software* más eficiente (RG Figueroa, 2008).

***Rational Unified Process* (RUP)**

La metodología RUP (Molpeceres, 2002), llamada así por sus siglas en inglés *Rational Unified Process*, es una de las más utilizadas para el análisis, implementación y documentación de sistemas orientados a objetos, se divide en 4 fases el desarrollo del *software*:

- Inicio, en esta etapa se determina la visión del proyecto.
- Elaboración, su objetivo es determinar la arquitectura óptima.

- Construcción, es la etapa en la que se logra obtener la capacidad operacional inicial.
- Transmisión, el objetivo de esta etapa es llegar a obtener el lanzamiento del proyecto.

Cada una de estas faces es desarrollada mediante el ciclo de iteraciones, el cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes (Y Araujo, 2011).

Disciplina de Desarrollo

Ingeniería de Negocios: Entendiendo las necesidades del negocio.

Requerimientos: Trasladando las necesidades del negocio a un sistema automatizado.

Análisis y Diseño: Trasladando los requerimientos dentro de la arquitectura de *software*.

Implementación: Creando un *software* que se ajuste a la arquitectura y que tenga el comportamiento deseado.

Pruebas: Asegurándose que el comportamiento requerido es el correcto y que todos los requerimientos solicitados están presentes.

Disciplina de Soporte

Configuración y administración del cambio: Guardando todas las versiones del proyecto.

Administrando el proyecto: Administrando horarios y recursos. Ambiente: Administrando el ambiente de desarrollo.

Distribución: Garantizar la salida del proyecto.

Los elementos del RUP son:

Actividades, son los procesos que se determinan en cada iteración.

Trabajadores, son las personas o entes involucrados en cada proceso.

Artefactos, un artefacto puede ser un documento, un modelo, o un elemento de modelo.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del *software*.

1.5.2 Metodologías ágiles

Los procesos de desarrollo ágiles de *software* están diseñados para producir *software* de manera rápida. La metodología ágil constituye un proceso interactivo en el que se enlazan la especificación, el diseño, el desarrollo y las pruebas.

SCRUM

Define un marco para la gestión de proyectos. Está especialmente indicada para proyectos con un rápido cambio de requisitos. El desarrollo de *software* se realiza mediante iteraciones, denominadas *sprints*, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto, el equipo de desarrollo debe reunirse con habitualidad para coordinación e integración del *software* (K Schwaber, 2002).

Ventajas de utilizar SCRUM como metodología ágil de desarrollo de *software* (J Palacio, 2010):

- Cumplimiento de expectativas: el cliente establece sus expectativas indicando el valor que aporta a cada requisito/historia del proyecto, el equipo los estima y con esta información el propietario del producto establece su prioridad.
- Flexibilidad a cambios: genera una alta capacidad de reacción ante los cambios de requerimientos generados por necesidades del cliente o evoluciones del mercado. La metodología está diseñada para adaptarse a los cambios de requerimientos que conllevan los proyectos complejos.
- Reducción del tiempo: el cliente puede empezar a utilizar las funcionalidades más importantes del proyecto antes de que esté finalizado por completo.
- Mayor calidad del *software*: la forma de trabajo y la necesidad de obtener una versión funcional después de cada iteración ayuda a la obtención de un *software* de calidad superior.
- Mayor productividad: se consigue entre otras razones, gracias a la eliminación de la burocracia y a la motivación del equipo que proporciona el hecho de que sean autónomos para organizarse.

- Maximiza el retorno de la inversión (ROI): producción de *software* únicamente con las prestaciones que aportan mayor valor de negocio gracias a la priorización por retorno de inversión.
- Predicciones de tiempos: mediante esta metodología se conoce la velocidad media del equipo por sprint (los llamados puntos historia), con lo que consecuentemente, es posible estimar fácilmente el tiempo en el que se dispondrá de una determinada funcionalidad que todavía está retrasada.

Programación extrema (*eXtreme Programming* (XP))

Es una metodología de desarrollo de *software* ágil (Beck, 2008), que considera a las personas como un factor decisivo para lograr el éxito de un proyecto. Por ser un proceso ágil tiene como principal característica su adaptación a entornos cambiantes. Esto es posible porque el proceso está diseñado para adaptarse en forma inmediata a los cambios, con bajos costos asociados en cualquier etapa del ciclo de vida. Es una de las metodologías más exitosas en la actualidad, utilizada para proyectos de corto plazo con un equipo pequeño. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo al usuario final, facilitando de esta forma la toma de decisiones.

La metodología está basada en (Beck, 2008) (Molpeceres, 2002):

Pruebas Unitarias: son pruebas realizadas a los principales procesos, de tal forma que se puedan realizar la validación de *software*, a lo largo de todo el proceso de desarrollo.

Prefabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares.

Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo.

XP propone que el desarrollo de *software* comience a una menor escala y añada funcionalidades con retroalimentación continua. El manejo del cambio se convierte en parte sustantiva del proceso. No introduce funcionalidades antes que sean necesarias. El cliente o usuario se convierte en miembro del equipo.

La metodología XP divide su ciclo de vida en fases:

Fase I: Exploración

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología (Beck, 2008).

Fase II: Planificación de la Entrega

En esta fase el cliente establece la prioridad de cada historia de usuario, y mientras, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses (Beck, 2008).

Fase III: Iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo certeramente historias de usuario que fuercen la creación de una determinada arquitectura, sin embargo, no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción (Beck, 2008) (Molpeceres, 2002).

Fase IV: Producción

La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase (Beck, 2008).

Fase V: Mantenimiento

Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura (Beck, 2008).

Fase VI: Muerte del Proyecto

En esta fase el cliente no cuenta con nuevos requisitos funcionales para añadir al sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad. Se genera la documentación y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando el presupuesto se agota, imposibilitando mantenerlo activo (Beck, 2008).

1.5.3 Selección de la metodología de *software* a utilizar

Después de analizar las metodologías de desarrollo de *software*: RUP, XP y SCRUM, se puede afirmar:
RUP es una de las más robustas metodologías para el análisis, diseño, implementación, soporte y documentación de sistemas informáticos orientados a objetos. El levantamiento exhaustivo de requerimientos, la realización del análisis y diseño detallados, y la serie de artefactos y roles que propone demandan de un gran equipo de trabajo. En este caso se cuenta con un grupo de trabajo reducido por lo cual se dificulta el cumplimiento óptimo de estos procesos, por lo cual se hace necesario la utilización de una metodología ágil.

SCRUM es una metodología diseñada para proyectos con un cambio repentino en los requisitos, se enfoca en prácticas de organización y gestión. La poca documentación que esta maneja puede provocar efectos negativos irremediables en la confección de software.

XP por su parte define grupos de trabajo reducidos. Centra su prioridad en las necesidades del cliente satisfaciéndolo mediante un desarrollo iterativo e incremental, abierto a cambios y caracterizado por un código simple. El equipo de trabajo incluye al usuario final por lo cual este está disponible para responder preguntas correctamente y fijar prioridades, de forma que no atrase la toma de decisiones, facilitado así la

retroalimentación y una mejor calidad del producto final. Por estas razones se ha seleccionado para la realización de este trabajo.

1.6 Herramientas y tecnologías

Las herramientas y tecnologías que se seleccionaron para el desarrollo del componente de cifrado biométrico, tienen el objetivo de facilitar, flexibilizar y organizar dicho proyecto.

1.6.1 Herramientas CASE

Las herramientas CASE (Ingeniería de *Software* Asistida por Computadora, en inglés *Computer Aided Software Engineering*) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de *software* reduciendo el coste de las mismas en términos de tiempo y finanzas (LÓPEZ, 2012).

Visual Paradigm for UML 8.0

Herramienta profesional que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue (para metodologías de ciclo completo). El *software* de modelado UML agiliza la construcción de aplicaciones de calidad. Propicia además un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación hasta el análisis y el diseño, además permite la correcta realización de los artefactos propuestos por la metodología (LÓPEZ, 2012).

Se seleccionó la herramienta CASE Visual Paradigm for UML 8.0 pues soporta el ciclo completo del desarrollo de *software*. Es una herramienta gratuita y además se encuentra disponible para distribución Linux. Permite el uso de un lenguaje estándar que facilita la comunicación entre todo el equipo de desarrollo.

1.6.2 Lenguaje de Programación

Un lenguaje de programación es un lenguaje formal que permite al programador comunicarse de forma directa con la máquina, a través de algoritmos y reglas lógicas. Estos están formados por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

Java

Es un lenguaje de programación concurrente orientado a objetos, desarrollado por Sun Microsystems. Se deriva de C y C++, eliminando parte de las complejidades que muestran estos lenguajes, es independiente de plataforma, es ejecutado sobre una máquina virtual que le permite no verse afectado por la arquitectura de la computadora (Meza, 2013).

Java cuenta con procedimientos que garantizan la seguridad durante la ejecución, no viola las restricciones de seguridad del sistema donde se ejecute, este gestor de seguridad que posee permite controlar el acceso de la aplicación a los recursos del sistema. Su licencia es bajo la Licencia Pública General de GNU, dándole más popularidad frente a otros lenguajes.

C#

C# (Groussard, 2012) es un lenguaje orientado a objetos simple, elegante y con seguridad que permite generar una gran variedad de aplicaciones. Proporciona mecanismos intrínsecos de código de confianza para obtener un alto nivel de seguridad, la recolección de elementos no utilizados y la seguridad de tipos. Admite herencia única y crea lenguaje intermedio de Microsoft como entrada de compiladores de código nativo.

Está completamente integrado con .NET Framework y *Common Language Runtime*, que conjuntamente proporcionan interoperabilidad del lenguaje, recolección de elementos no utilizados, seguridad ampliada y compatibilidad de versiones mejorada. C# simplifica y moderniza algunos de los aspectos más complejos de C y C++, como los espacios de nombres, las clases, las enumeraciones, la sobrecarga y el control estructurado de excepciones. Elimina ciertas características de C y C++ como macros y herencia múltiple. Para los programadores de C++ actuales, C# proporciona un lenguaje alternativo de gran potencia y productividad.

C++

Hacia el año 1979 se crea por *Bjarne Stroustrup* (Lorente,1993) una versión experimental con la intención de proporcionar una herramienta de desarrollo para el núcleo UNIX en ambientes distribuidos. Ya en 1983 se bautiza como C++ y en 1985 *Stroustrup* publica la primera edición del libro "*The C++ Programming Language*" que sirvió de estándar informal y texto de referencia del mismo. C++ incorpora muchas

características, tales como: Programación Orientada a Objetos, tratamiento de excepciones, sobrecarga de operadores y plantilla.

Como lenguaje C, C++ adopta una visión muy cercana al lenguaje máquina. El lenguaje está formado por instrucciones muy explícitas, cortas, cuya duración de ejecución puede preverse con antelación, en el momento de escribir el programa.

Los programas hechos en C++ son rápidos, portables, flexibles y versátiles. Tiene características de lenguaje de alto y bajo nivel. C++ es un lenguaje de programación de propósito general que ofrece economía sintáctica, control de flujo, estructuras sencillas y un buen conjunto de operadores. Por estas razones se decidió utilizar como lenguaje de programación para el desarrollo del componente de cifrado biométrico.

1.6.3 Entorno de Desarrollo Integrado (IDE)

Se define como IDE (*Integrated Development Environment*) o entorno de desarrollo integrado en español el programa compuesto por un conjunto de herramientas para un programador, es decir, consiste en un editor de código, un compilador, un depurador y en algunos casos un constructor de interfaz gráfica. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien, poder utilizarse para varios de ellos (MARTÍNEZ, 2007).

Para la realización de este trabajo se decidió emplear el IDE Qt Creator en su versión 3.3.0, debido a que utiliza el lenguaje de programación C++ de forma nativa y la plataforma de desarrollo es el Framework Qt. Qt Creator es un entorno integrado de desarrollo bastante completo, moderno, potente, fácil de manejar, eficiente, abierto y gratuito, que permite el desarrollo rápido de aplicaciones en entornos MS Windows, Mac OS y Linux.

Plataforma de desarrollo

Para el desarrollo del componente se utilizará el Framework Qt en su versión 5.7.0, es un *framework* de interfaz de usuario que se utiliza para la creación de dispositivos y el desarrollo de aplicaciones de soporte a más de una docena de plataformas líderes.

Conclusiones parciales

El análisis realizado sobre los diferentes aspectos y conceptos involucrados en el proceso de cifrado de plantillas de minucias de huellas dactilares posibilitó un mejor entendimiento del contexto de la investigación y de la problemática a resolver. El estudio de las características y funcionamiento de algunos de los métodos utilizados permitieron la certera selección de método a implementar. Como resultado del análisis de las peculiaridades de las metodologías de desarrollo, herramientas y tecnologías existentes se seleccionaron aquellas que apoyarán el ciclo de vida de la solución.

CAPÍTULO 2: DESCRIPCIÓN DEL DESARROLLO DEL COMPONENTE

Introducción

En el presente capítulo se realiza una descripción del desarrollo del componente, detallando los patrones que se utilizan para modelar el diseño de la solución y una breve explicación y ejemplificación de los artefactos utilizados para el desarrollo.

2.1 Propuesta de la herramienta a desarrollar

El componente a desarrollar está basado en el modelo de plantillas cancelables propuesto en (Nalini K. Ratha, 2007) Estará compuesto por la implementación de un algoritmo que realizará el proceso de cifrado y otro algoritmo que realizará el proceso de comparación en el dominio protegido.

Cifrado mediante transformación cartesiana

Como primer paso para realizar el algoritmo de cifrado (según Nalini K. Ratha, 2007), una vez obtenida y leída la plantilla de minucias, se ubican en un sistema de coordenadas cartesianas, como referencia el núcleo y alineando el eje de las abscisas con su orientación. Este sistema de coordenadas será dividido en celdas con un tamaño fijo que posteriormente cambiarán de posición mediante las especificaciones de la clave generada aleatoriamente, mientras las minucias dentro de las celdas conservan sus posiciones relativas.

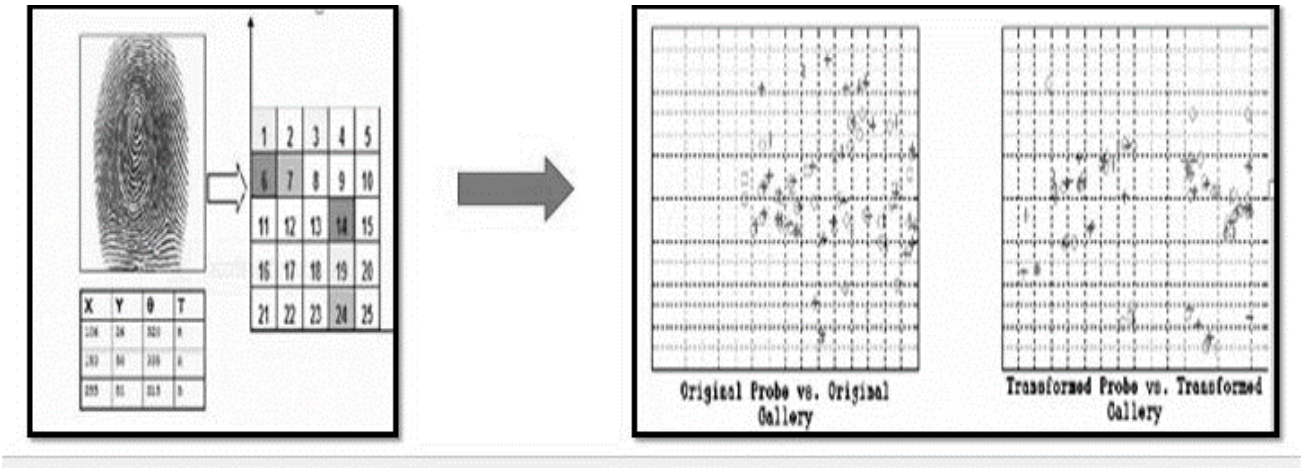


Figura 7. "Ejemplo de cifrado de plantilla biométrica mediante transformación cartesiana"

Comparación

Para realizar el proceso de comparación, una vez obtenida la plantilla de minucias de consulta, el sistema genera las estructuras triangulares correspondientes y realiza la búsqueda de al menos una coincidencia entre estas y el conjunto de estructuras triangulares de la plantilla cifrada (Jeffers, 2007). En caso de existir se alinean, aplicando rotación y traslación, al conjunto de puntos que representan las minucias de consulta para hacerlos corresponder. De lo contrario, en caso de no encontrarse coincidencia, se asume que la plantilla de minucias de consulta no corresponde al rasgo biométrico protegido.

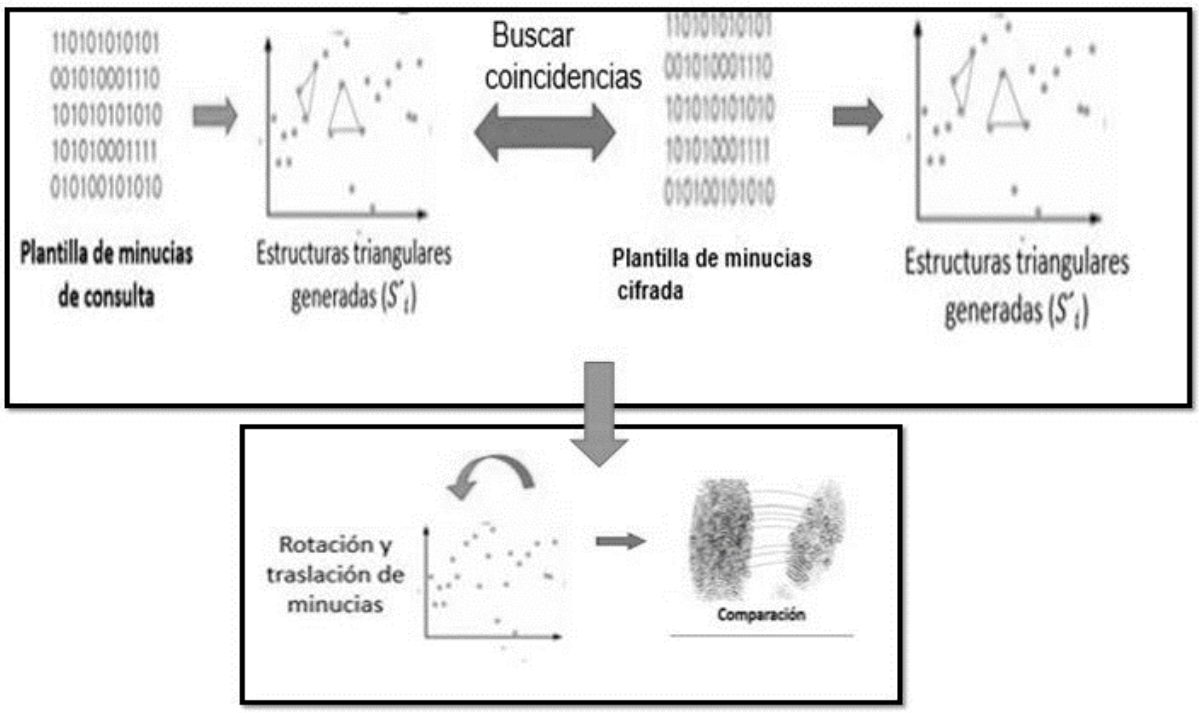


Figura 8. "Proceso de comparación de plantillas de minucias en el dominio cifrado"

2.2 Modelo de dominio

El Modelo de dominio es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema, conceptos del mundo real, no de los componentes de *software*. Una clase conceptual puede ser una idea o un objeto físico (símbolo, definición y extensión).

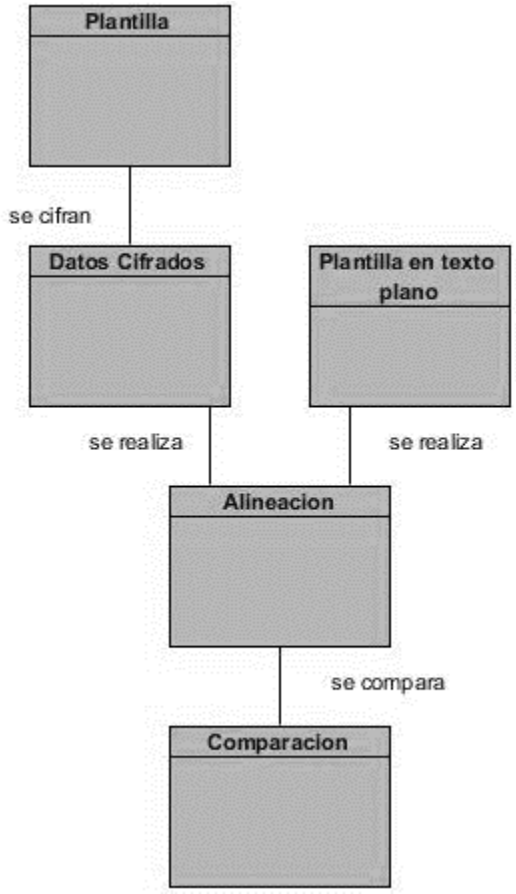


Figura 9. "Modelo de dominio"

Modelo de dominio	Propósito
Plantilla	La plantilla es el archivo que contiene los datos biométricos a cifrar.
Plantilla en texto plano	Es una plantilla de consulta usada para la alineación y comparación en el dominio protegido.

Datos Cifrados	Estos son los datos provenientes de una plantilla, ya cifrados mediante el método plantillas cancelables utilizando como tipo de transformación la trasformación cartesiana.
Comparacion	Brinda la respuesta final según la correspondencia biométrica.
Alineacion	Genera las estructuras de minucias del cifrado y la plantilla de consulta y alinea las estructuras generadas para la comparación.

Tabla 1. "Descripción del modelo de dominio"

2.3 Fase de planificación

La metodología XP define a la planificación como la etapa inicial de todo proyecto. Es aquí donde comienzan a interactuar los miembros del grupo de desarrollo para determinar los requisitos que deberá cumplir el componente a implementar y se identifican el número y tamaño de las iteraciones, así como otros aspectos de interés.

2.3.1 Captura de requisitos

La captura de requisitos es uno de los procesos más importantes para una correcta realización en las etapas de desarrollo del componente. Un error en las fases iniciales puede repercutir en el incumplimiento de las funciones para las cuales se implementa, por lo que resulta significativo la obtención de los requisitos tanto funcionales como no funcionales.

2.3.2 Requisitos funcionales (RF)

RF1: Cargar la plantilla de minucias.

RF1.1: Leer la plantilla de minucias.

RF2: Cifrar los datos.

RF3: Alinear la plantilla de minucias.

RF4: Comparar las plantillas en el dominio cifrado.

2.3.3 Requisitos no funcionales (RNF)

1.*Software*

- Multiplataforma.
- *Framework* de desarrollo: Qt 5.7.0

2. *Hardware*

- PC Dual Core o superior.
- CPU 1.3 GHZ o superior.
- 1 Gb de memoria RAM o superior.

3. Restricciones en el diseño y la implementación

- Lenguaje de programación: C++.
- IDE: Qt Creator 3.3.0.
- Para el modelado de UML: Visual Paradigm 8.0.

2.3.2 Historias de usuario

Las historias de usuarios (HU) son la técnica utilizada para especificar los requisitos del *software*. Se trata de tarjetas en las cuales se describe brevemente las características que el sistema debe poseer. El tratamiento de las HU es muy dinámico y flexible, cada una de ellas es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas (José H. Canós, 2010)

Historia de Usuario	
Número: HU_1	**Usuario:** Sistema
Nombre historia: Cargar la plantilla de minucias.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1

Programador responsable: Laura Elena González Varela
Descripción: El sistema solicita la dirección del fichero a cargar, la cual es especificada por el usuario.
Observaciones: -

Tabla 2. "Historia de usuario correspondiente a la funcionalidad Cargar la plantilla de minucias"

2.3.3 Plan de iteraciones

Una vez descritas e identificadas las historias de usuarios y estimado el esfuerzo propuesto para la realización de cada una de ellas, se procede a la planificación de la etapa de implementación del sistema.

Se definieron 2 iteraciones para el desarrollo del sistema, las cuales se detallan a continuación:

Iteración 1: En esta iteración se realizarán las historias de usuario que inciden en el cifrado de los datos biométricos mediante el método plantillas cancelables.
Iteración 2: En esta iteración se realizarán las historias de usuario que propician la comparación de los datos biométricos en el dominio protegido. Una vez concluida esta iteración el componente se encontrará completamente concluido y listo para su explotación.

2.3.4 Plan de duración de las iteraciones

El plan de duración de las iteraciones es el encargado de mostrar las HU de acuerdo al orden de implementación en cada iteración, así como la duración estimada de estas. La siguiente tabla muestra la duración estimada en semanas por cada iteración:

Iteración	HU	Estimación (Semanas)
1	Cargar la plantilla de minucias	6
	Leer plantilla de minucias	
	Cifrar los datos mediante la transformación cartesiana	
	Cifrar los datos mediante la transformación polar	

	Alinear la plantilla de minucias	
2	Comparar en el dominio protegido	8

Tabla 3. ¨Tabla de iteraciones¨

2.6 Diseño del sistema

Para el diseño de aplicaciones informáticas la metodología XP no requiere la presentación del sistema mediante diagramas de clases utilizando notación UML, en su lugar se usan otras técnicas como las tarjetas CRC (Clase, Responsabilidad y Colaboración). El uso de estos diagramas puede aplicarse siempre y cuando mejoren la comunicación, tratando que no sean un peso en su mantenimiento, no sean extensos y que enfoquen la información de mayor importancia (Extreme Programing, 2009).

2.6.1 Tarjetas CRC

Para poder diseñar el sistema se debe cumplir con tres principios: Cargo o Clase, Responsabilidad y Colaboración (CRC). Las tarjetas CRC permiten desprenderse del método de trabajo basado en procedimientos y trabajar con una metodología basada en objetos. Las tarjetas CRC además permiten que el equipo completo contribuya en la tarea del diseño. Una tarjeta CRC representa un objeto, el nombre de la clase se coloca a modo de título en la tarjeta, las responsabilidades se colocan a la izquierda, y las clases que se implican en cada responsabilidad a la derecha, en la misma línea que su requerimiento correspondiente.

Nombre de las clases: Es cualquier persona, evento, concepto, pantalla o reporte.

Responsabilidades: Las responsabilidades de una clase son aquello que se conoce que la misma realiza, sus atributos y métodos.

Colaboradores: Los colaboradores de una clase son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades.

A continuación, se encuentra representada la tarjeta CRC correspondiente a la clase Mainwindow. Las tarjetas CRC de cada una de las clases se encuentran descritas en los anexos (Anexo 2).

Nombre de la clase: Mainwindow

Responsabilidades	Colaboradores
CargarPlantilla	1. Minucia
LeerPlantilla	1. Minucia

Tabla 4. ¨Tarjeta CRC correspondiente a la clase Mainwindow¨

2.6.2 Arquitectura del sistema

En su forma más sencilla, la arquitectura es la estructura de organización de los componentes de un programa (módulos), la forma en la que estos interactúan y la estructura de datos que utilizan. Sin embargo, en un sentido más amplio, los componentes se generalizan para que representen los elementos de un sistema de mayor alcance y sus interacciones (Pressman, 2003)

Para el desarrollo del producto se utilizará una **arquitectura basada en componentes** ya que dicha arquitectura brinda la posibilidad de reutilizar los componentes de forma independiente en otras soluciones y ofrece una descripción del esqueleto estructural y general de la aplicación (Camacho, y otros, abril 2004). Una arquitectura basada en componentes describe una aproximación de ingeniería de *software* al diseño y desarrollo de un sistema. Este patrón pone énfasis en la descomposición del sistema en componentes lógicos o funcionales y define una aproximación de diseño que usa componentes discretos, los que se comunican a través de interfaces que contienen métodos, eventos y propiedades. Esto provee un nivel de abstracción mayor que los principios de orientación por objetos y no se enfoca en asuntos específicos de los objetos como los protocolos de comunicación y la forma como se comparte el estado (Camacho, y otros, abril 2004).

El sistema a implementar consta de cinco componentes. Un componente visual que permite una vista a grandes rasgos del proceso, además de ser el responsable de cargar los datos necesarios para realizar el cifrado y la comparación. El componente de cifrado, que es donde se transforman los datos mediante la transformación cartesiana y la trasformación polar. El componente comparación que brinda la respuesta final según la correspondencia biométrica. Un componente denominado utililes que agrupa las clases que son comúnmente utilizadas por el resto de los componentes. Un componente alineación el cual interviene directamente en la comparación de la plantilla original con la plantilla cifrada.

En (Pressman, 2010) los patrones arquitectónicos son restricciones que definen como se integran los componentes para formar el sistema o modelos que permiten comprender las propiedades de un sistema general en función de las propiedades conocidas de las partes que lo integran.

El patrón arquitectónico propuesto para el desarrollo del componente es **Tuberías y Filtros**, pues provee una estructura para los sistemas que procesan un flujo de datos. Puede ser descompuesto en diferentes etapas, donde cada una es sucesiva e incremental con respecto a la anterior. Cada paso de procesamiento está encapsulado en un componente filtro. El dato pasa a través de tuberías entre filtros adyacentes (Camacho, y otros, abril 2004).

En el patrón arquitectónico **Tuberías y Filtros**, una tubería (*pipeline*) es una arquitectura que conecta componentes computacionales (filtros) a través de conectores (*pipes*), de modo que los procesos se ejecutan como un flujo de los mismos. Los datos se transportan a través de las tuberías entre los filtros, transformando gradualmente las entradas en salidas, hasta que son asignadas a un destino final: salida o almacenamiento (Reynoso,2004).

El componente a implementar consta de dos algoritmos (tuberías), los cuales se dividen en diferentes pasos que se ejecutan secuencialmente y cada filtro consume y procesa sus datos de forma incremental.

Cifrado

1. Entrada: Plantilla de minucias cargada.
2. Filtro 1: Ubicación de las minucias en un sistema de coordenadas cartesianas.
3. Filtro 2: Generación de la clave.
4. Filtro 3: Cifrado de la plantilla de minucias.
5. Salida: Plantilla de minucias cifrada.
6. Plantilla de minucias cifrada.

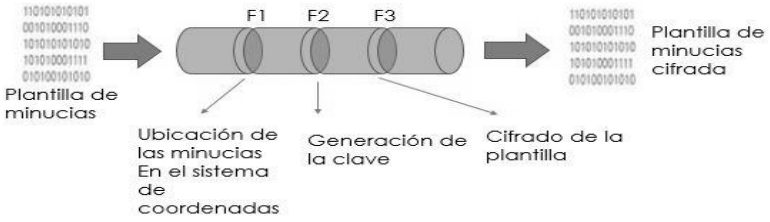


Figura 10. "Patrón tuberías y filtros en el proceso de cifrado de los datos"

Comparación

7. Entrada: Plantilla de minucias de consulta cargada.
8. Filtro 1: Formación de las tripletas de minucias.
9. Filtro 2: Determinación de las características identificativas de las tripletas.
10. Filtro 3: Búsqueda de coincidencias entre las características de las tripletas de la plantilla cifrada y el de la plantilla original.
11. Filtro 4: Rotación y traslación de las minucias de consulta.
12. Filtro 6: Comparación.
13. Salida: Respuesta que brinda el sistema sobre la autenticidad de la huella dactilar.

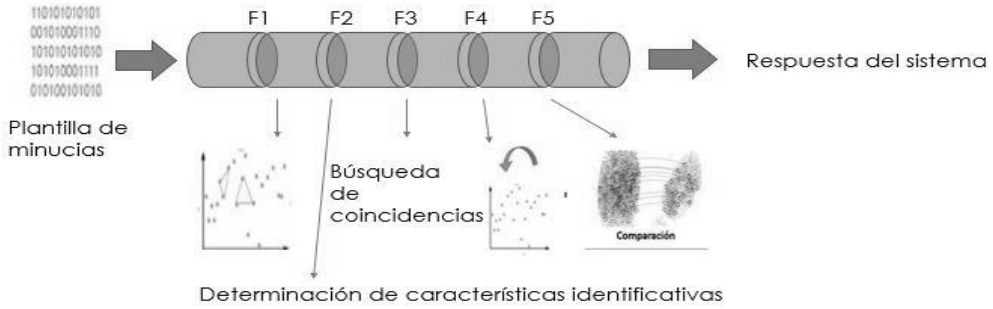


Figura 11."Patrón tuberías y filtros en el proceso de comparación de los datos"

2.7 Patrones de diseño

Un patrón de diseño es una forma conveniente para la reutilización de código orientado a objetos entre los proyectos y entre los programadores. La idea detrás de los patrones de diseño es simple: para catalogar comunes interacciones entre los objetos. Estos se han convertido en un elemento básico del diseño orientado a objetos y la programación, los mismos proporcionan soluciones elegantes, fáciles de reutilizar y de mantener (Cooper, 2000).

Los patrones de diseños que se evidencian en la implementación están dentro de la Banda de Cuatro (GOF por sus siglas en inglés) y en los Patrones de *Software* para la Asignación General de Responsabilidad (GRASP).

2.7.1 Patrones GRASP

Los patrones GRASP definen básicamente que clase se encarga de cada función en instante dado. Para la modelación de la solución se han tenido en cuenta los siguientes patrones GRASP (Mora, 2003-2004).

- **Creador:** Este patrón define qué clase crea qué objeto, para ello la clase creadora debe contener, agregar, componer, registrar, tener los datos de inicialización o utilizar muy de cerca otra clase. Se emplea en la clase Cifrado, pues este patrón es el encargado de identificar quién es el responsable de la creación de los objetos.
- **Alta Cohesión**: Para que exista alta cohesión es necesario que cada elemento o clase en el diseño realice una única función en la solución, y que solo sea realizada por ella. Para lograr esto es necesario realizar una asignación adecuada de responsabilidades y evitar que las clases estén saturadas de métodos. Es utilizado en la clase Minucia, pues permite asignar la mayor cantidad de responsabilidades a la clase y que esta siga siendo sencilla
- **Bajo acoplamiento**: Este patrón se enfoca en el empleo de clases poco ligadas entre sí, de forma que en caso de ocurrir una modificación de último momento en la misma tenga la mínima repercusión en el resto de las clases. En la clase Triangulo se evidencia el uso de dicho patrón.
- **Experto en información**: Este patrón es utilizado en la clase Comparación, pues dada una situación, la información necesaria para proveer una solución se obtiene con facilidad. Los objetos presentes en ella se valen de su propia información para realizar la tarea asignada, conservando el encapsulamiento.
- **Controlador:** Asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, entre otros). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión (Mora, 2003-2004). Se emplea en las clases Cifrado y Comparación, pues este patrón plantea el uso de determinadas clases que sean las encargadas de dirigir todo el proceso.

2.7.2 Patrones GOF

Los patrones GOF (Cooper, 2000) describen soluciones simples y de una manera elegante a problemas en un contexto particular, dentro del diseño de software orientado a objetos. En la modelación del sistema se han tenido en cuenta el siguiente patrón GOF:

- ***Prototype:*** Este patrón permite la creación de nuevos objetos clonándolos de una instancia ya existente, copiando el prototipo de esa clase. Se evidencia el uso de este en la clase Mainwindow a la hora de crear ventanas de diálogo.

Conclusiones parciales

Durante el desarrollo del capítulo, se analizaron los procesos que conforman el componente para un mejor entendimiento del mismo. La definición del modelo de dominio permitió la correcta identificación de los requisitos funcionales del sistema. Las historias de usuario correspondientes a cada uno de los requisitos permitieron la adecuada comprensión de sus especificaciones. Mediante el uso de la arquitectura basada en componentes, apoyada en el patrón arquitectónico tuberías y filtros y en el uso de patrones GRASP y GOF, se logró una mejor organización de la propuesta solución, sentando bases en la implementación del componente de cifrado de minucias de huellas dactilares.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS DEL COMPONENTE

Introducción

Una vez diseñado el sistema y definida la arquitectura, se procede a la implementación del componente dando cumplimiento a los requisitos planteados. En este capítulo se definen los criterios específicos a utilizar para un mejor entendimiento del código. Se diseña el diagrama de componentes, el cual posibilita la comprensión del flujo de trabajo de la implementación. Se realizan un conjunto de pruebas para validar el correcto funcionamiento del componente implementado y su correspondencia con los requisitos funcionales definidos.

3.1 Estándares de codificación

La implementación se realizará bajo estándares de codificación para lograr una mejor comprensión del código. Los estándares de codificación son de gran importancia debido a las siguientes razones:

- ✓ El 80% del coste del código de un programa va a su mantenimiento.
- ✓ Casi ningún *software* es mantenido en toda su vida útil por el autor original.
- ✓ Las convenciones de código mejoran la lectura del *software*, permitiendo entender código nuevo mucho más rápidamente y más a fondo.
- ✓ Si se distribuye el código fuente como un producto, es necesario asegurarse de que está bien hecho y presentado como cualquier otro producto.

Reglas de codificación

1. Cada línea de código debe contener solo una sentencia.
2. Cada método implementado debe tener un comentario previo de su funcionamiento.
3. El código fuente debe ser en su totalidad escrito en español.
4. Se debe evitar las líneas de más de 85 caracteres, pues no son bien manejadas por muchas herramientas.
5. Se debe seguir las convenciones de nombres mostradas en la tabla 5.

Tipos de identificadores

- **Variables:** los nombres de las variables comenzarán con minúscula, en caso de ser un nombre compuesto la inicial de la primera palabra en minúscula y la de las otras palabras que lo componen en mayúscula.

```
QList<Minucia> listadeMinucia;
int cantidadDeMinucia;
```

Figura 12 "Ejemplo del uso de estándares de codificación en variables"

- **Métodos:** los nombres de los métodos deberán sugerir la acción a desarrollar y comenzarán con mayúscula. En caso de contener más de una palabra, estas irán unidas y comenzarán también con mayúscula.

```
QList <Minucia>* MainWindow::LeerPlantilla(const QString &fileName)
```

Figura 13 "Ejemplo del uso de estándares de codificación en métodos"

- **Clases o interfaces:** los nombres de las clases tendrán la primera letra mayúscula, y si es compuestos la primera letra de cada palabra en mayúscula, nombres simples y de alguna manera que describa el contenido, usar palabras completas, a no ser que la abreviatura sea muy conocida.

```
class Plantilla
{
```

Figura 14"Ejemplo del uso de estándares de codificación en clases o interfaces"

- **Constantes:** cada letra que pertenezca al nombre de la constante se escribirá con mayúscula y en caso de ser un nombre compuesto, se separarán por un guión bajo "_".

3.2 Tareas de ingeniería

De las HU se derivan una o varias tareas de ingeniería, estas son pasos lógicos a seguir por los desarrolladores. Cada tarea tiene relacionada una HU, tiempo estimado, programador asignado y una descripción breve de la misma.

Iteración 1	
Historias de Usuario	**Tareas**

Cargar la plantilla de minucias	1. Búsqueda del directorio donde se encuentra la plantilla de minucias. 2. Obtención de la plantilla para el procesado.
Cifrar los datos	1. Creación de la coordenada. 2. Definición de los cuadrantes. 3. Introducción de cada minucia en su cuadrante correspondiente 4. Generación de la clave. 5. Multiplicación de la matriz clave y la matriz coordenada. 6. Obtención de la plantilla cifrada.
Almacenar los datos cifrados	1. Pedido de la dirección donde se almacenarán los datos cifrados.

Tabla 5. "Desglose de las tareas de ingeniería correspondientes a la primera iteración"

Las tareas de ingeniería correspondientes a la segunda iteración se encuentran anexadas al documento (Anexo 3).

3.3 Diagrama de componentes

El diagrama de componentes permitirá un mejor entendimiento del flujo de trabajo de la implementación. El mismo muestra los componentes de *software* que se emplearán en la construcción de la solución y las relaciones existentes entre ellos, así como sus dependencias, ubicación y otros aspectos de interés.

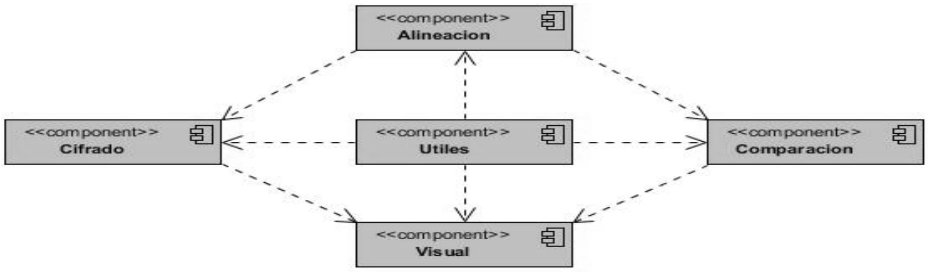


Figura 15 "Diagrama de componentes"

Componente	Propósito
Visual	Permite una vista de las principales funcionalidades del componente, además de ser el responsable de cargar los datos de la plantilla para realizar posteriormente el cifrado y la comparación.
Cifrado	Cifrar la plantilla mediante el método plantillas cancelables utilizando como tipo de transformación la trasformación cartesiana.
Comparacion	Brinda la respuesta final según la correspondencia biométrica.
Utiles	Agrupa las clases que son utilizadas por el resto de los componentes.
Alineacion	Genera las estructuras de minucias del cifrado y la plantilla de consulta y alinea las estructuras generadas para la comparación.

Tabla 6. "Descripción de los componentes"

3.4 Pruebas

El proceso de pruebas o validación de un sistema, es una de las etapas fundamentales del desarrollo del mismo. Es donde se valida la calidad y la aceptación que tiene la herramienta para el cliente.

Cualquier pieza de *software* completo, desarrollado o adquirido, puede verse como un sistema que debe probarse, ya sea para decidir acerca de su aceptación, para analizar defectos globales o para estudiar aspectos específicos de su comportamiento (Pressman, 2010).

3.4.1 Pruebas de validación

Las pruebas de validación se enfocan en las acciones por parte del usuario y las respuestas por parte del sistema. Se llevan a cabo para comprobar los requerimientos y se considera que una funcionalidad tiene éxito cuando se comporta de la manera esperada por el cliente (Pressman, 2010).

Técnica de caja negra

Las pruebas de caja negra son las que se aplican a la interfaz del *software*. Una prueba de este tipo examina algún aspecto funcional de un sistema que tiene poca relación con la estructura lógica interna del *software* (Pressman, 2010).

Las pruebas de caja negra, también denominadas pruebas de comportamiento, se concentran en los requisitos funcionales del sistema. Es decir, permiten al ingeniero derivar conjuntos de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales de un programa. Las pruebas de caja negra tratan de encontrar errores en las siguientes categorías:

- ✓ Funciones incorrectas o faltantes.
- ✓ Errores de interfaz.
- ✓ Errores de estructuras de datos o en acceso a bases de datos externas.
- ✓ Errores de comportamiento o desempeño.
- ✓ Errores de inicialización y término.

A continuación, se muestran los casos de pruebas correspondientes a las funcionalidades Cargar la plantilla de minucias y Almacenar la plantilla cifrada:

Casos de prueba	
Código Caso de Prueba: 1	**Nombre Historia de Usuario:** Cargar la plantilla de minucias.
Nombre de la persona que realiza la prueba: Laura Elena González Varela**.**	
Descripción de la prueba: Buscar el directorio donde se encuentra la plantilla de minucias para posteriormente cargarla con vistas al procesado de esta.	

Condiciones de ejecución: La plantilla debe cumplir con la estructura correspondiente.
Entrada/Pasos de ejecución: Nombre del fichero, que incluye la dirección.
Resultado esperado: La plantilla de minucias cargada.
Evaluación: Satisfactoria

Tabla 7. "Tabla de casos de prueba correspondiente a la funcionalidad Cargar plantilla"

Casos de prueba	
Código Caso de Prueba: 2	**Nombre Historia de Usuario:** Almacenar la plantilla cifrada.
Nombre de la persona que realiza la prueba: Laura Elena González Varela	
Descripción de la prueba: Una vez generada la plantilla cifrada se pedirá la dirección donde se almacenará.	
Condiciones de ejecución: La plantilla cifrada se almacenará en un fichero con extensión .pctc.	
Entrada/Pasos de ejecución: Dirección de almacenamiento.	
Resultado esperado: La plantilla cifrada almacenada.	
Evaluación: Satisfactoria	

Tabla 8. "Tabla de casos de prueba correspondiente a la funcionalidad Almacenar plantilla cifrada"

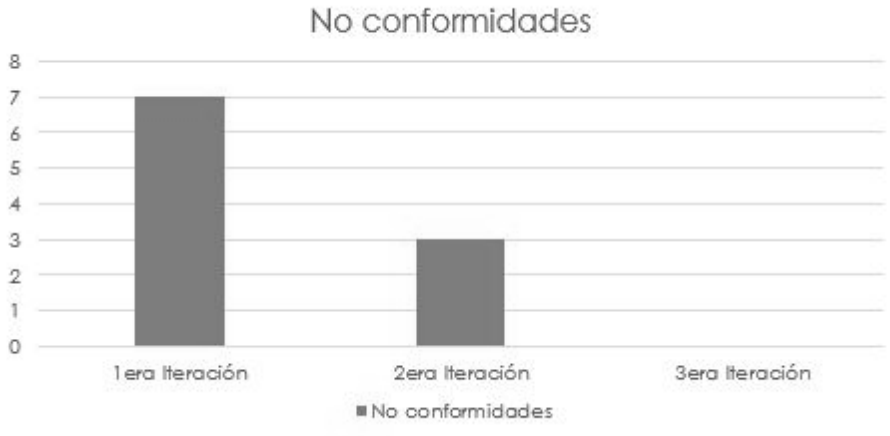


Figura 16. "Resultado de las pruebas de caja negra"

Se realizaron 3 iteraciones de estas pruebas, encontrándose en la primera iteración 7 no conformidades, en la segunda 3 y en la tercera ninguna. Entre las no conformidades identificadas estuvieron el insuficiente tratamiento de excepciones y la falta de claridad o ausencia de los mensajes de respuesta del sistema.

3.4.1 Pruebas de unidad

Las pruebas unitarias son pruebas definidas por los programadores. Estas pruebas son realizadas a pequeñas porciones de código por separados, para garantizar que determinado módulo satisfaga un comportamiento esperado antes de integrarse al sistema. Deben ser definidas antes de realizar el código y se pueden ir ejecutando a medida que se está implementando, no es indispensable esperar al final de la implementación del *software* (Peña, 2006).

Pruebas de caja blanca

Las pruebas de caja blanca derivadas a partir de las especificaciones internas del diseño o el código y requieren del conocimiento de la estructura interna del programa. Se examinan los caminos lógicos del sistema, definidos por condiciones y/o bucles. Se pueden examinar mediante ellas el estado del programa en varios puntos con el objetivo de determinar si el estado actual o real coincide con el esperado (Pressman, 2010).

Prueba del camino básico: Permite a la persona encargada de diseñar los casos de pruebas obtener una medida de la complejidad lógica de un diseño procedimental, con el fin de usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución (Letelier, 2009).

Los pasos que se siguen para aplicar esta técnica son:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.

2. Se calcula la complejidad ciclomática del grafo.

3. Se determina un conjunto básico de caminos independientes.

4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

La complejidad ciclomática de un grafo de flujo V(G) establece el número de caminos independientes.

La mencionada complejidad ciclomática se calcula de tres formas:

- V (G) = (A (Aristas) – N (Nodos)) + 2.
- V (G) = P (Nodos Predicados) + 1.
- V (G) = R (El número de regiones del grafo).

Funcionalidad: Combinaciones

```
int EstructuraMinucias::Combinaciones(){
        double nFac=1;/1
        double N_RFac=1;/1
        for (int i = 1; i <=Longitud; ++i) {/2
            nFac=i*nFac;/3
        }
        for (int i = 1; i <=(Longitud-3); ++i) {/4
            N_RFac=i*N_RFac;/5
        }
        double temp=(6*N_RFac);/6
        int devolver=int(nFac/temp);/6
        return devolver;/7
}
```

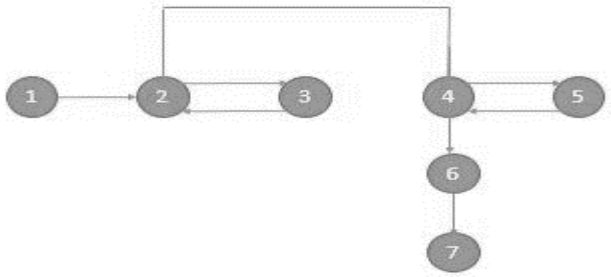


Figura 17. "Grafo perteneciente a la funcionalidad Combinaciones"

Complejidad ciclomática:

Fórmula 1:

V (G) = (A (Aristas) – N (Nodos)) + 2

V (G) = (8-7)+2=3

Fórmula 2:

V (G) = P (Nodos Predicados) + 1

V (G) = 2+1= 3

Fórmula 3:

V (G) = R = 3

Caminos independientes obtenidos:

1-2-3-2-4-5-6-7

1-2-4-5-6-7

1-2-3-4-6-7

3.4.1 Pruebas de efectividad

Rendimiento biométrico

Para estimar el desempeño del componente teniendo en cuenta las tasas de error se decide optar por la utilización una de las bases de datos más representativas a nivel internacional la FVC2000[10]. Las muestras almacenadas en cada una de ellas fueron tomadas con diferentes sensores:

BD1: Sensor óptico de bajo costo "*Secure Desktop Scanner*" por *Keytronic*.

BD2: Sensor capacitivo de bajo costo "*TouchChip*" por *ST Microelectronics*.

BD3: Sensor óptico "DF-90" por *Identicator Technology*.

BD4: Generador sintético de huellas digitales (SFinGe).

Las muestras de dichas bases de datos se caracterizan por tener una calidad entre baja y media, constituyendo esto un factor importante para ser elegidas como objeto de pruebas de rendimiento biométrico.

Las bases de datos están compuestas por 80 muestras de huellas dactilares, 8 muestras correspondientes a cada individuo, para un total de 10. Para la realización del proceso de alineación es necesaria la presencia de la singularidad núcleo en las muestras y debido a su ausencia en algunas no pudieron ser utilizadas todas las tomas contenidas en las bases de datos. Producto a esto se tipificaron las bases de datos, quedando constituidas por 60 imágenes de huellas dactilares distribuidas en 6 tomas diferentes por cada dedo de la mano.

Entrada: plantillas de minucias en texto claro y la plantilla cifrada para la comparación en el dominio protegido

Salida: taza de falso aceptado y taza de falso rechazo.

El cálculo de la FMR y la FNMR se realizó de la siguiente manera:

FMR = Probabilidad (D1/H0)

FNMR = Probabilidad (D0/H1)

Siendo:

D0: comparación negativa; D1: comparación positiva.

I: conjunto de consulta.

T: plantilla almacenada para comparación.

[10] **FVC2000:** Competencia Internacional de Algoritmos de Verificación de Huellas Dactilares.

H0: I distinto de T; H1: I igual a T. Donde la igualdad y la desigualdad representan la pertenencia o no al mismo rasgo biométrico.

En la tabla se pueden observar los resultados pertenecientes a las tazas de falso aceptado y faso rechazo por cada una de las bases de datos analizadas:

Base de datos del FVC 2000	FMR	FNMR
BD1	0.0375	0.3475
BD2	0.0125	0.3219
BD3	0	0.4976
BD4	0.0356	0.3457

Tabla 9. "Taza de FRR y FRA"

Seguridad biométrica

El análisis de seguridad criptográfica del componente de cifrado biométrico utilizando plantillas cancelables, se realiza mediante el cálculo de los bits de fortaleza que presenta la clave. Se analizan la cantidad de minucias que deben coincidir respecto al total de minucias para romper la transformación. Según (Ratha, 2001) (Fernández, 2016) para cada una de las transformaciones se realiza un análisis de seguridad criptográfica.

Para este análisis se utilizó una plantilla de 35 minucias, donde la cantidad mínima de minucias a comparar positivamente son 15, teniendo en cuenta que se codifican 8 bits por minucia. Utilizando la expresión para el cálculo de la fortaleza del componente.

$$p = 8m - \log_2 \binom{N}{m}$$

Figura 18 . "Fórmula para el cálculo de la seguridad criptográfica del componente de cifrado biométrico"

Como resultado de obtuvo que el método ofrece un valor de seguridad criptográfica de 66 bits

Conclusiones parciales

Durante el desarrollo del capítulo se definieron las pautas de codificación, teniendo como objetivo estandarizar el código para un mejor entendimiento del mismo por futuros desarrolladores y las tareas de ingeniería a realizar por cada historia de usuario para guiar en la implementación de los requerimientos de la solución. Se ejecutaron además las pruebas de validación al componente, corroborando el correcto funcionamiento del mismo y permitiendo verificar el cumplimiento de los requerimientos definidos por el cliente.

Conclusiones generales

- El análisis de la bibliografía referente a los diferentes modelos de protección de plantillas de minucias de huellas dactilares permitió la selección de un conjunto de algoritmos para realizar la protección de las plantillas de minucias utilizando el esquema de plantillas cancelables.
- El análisis de los diferentes enfoques propuestos en la bibliografía para el cifrado y la alineación de los datos facilitó la propuesta de un método de cifrado práctico que aumenta la seguridad de las plantillas de minucias de huellas dactilares utilizadas en el proceso de autenticación de personas.
- La elaboración de la propuesta de solución y la identificación de los requisitos del sistema a implementar permitieron la correcta confección de las historias de usuario, las cuales sirvieron de punto de partida en el desarrollo del componente.
- La definición de la arquitectura basada en componentes y el empleo del patrón arquitectónico tuberías y filtros posibilitaron la implementación de un componente eficiente, reutilizable y tolerante a fallos.
- Las pruebas realizadas permitieron determinar el correcto funcionamiento del componente y del esquema implementado.

Recomendaciones

Para mejorar el componente de cifrado biométrico se recomienda:

- Desarrollar otros algoritmos de selección de características identificativas para realizar la comparación de los datos.
- Desarrollar una estrategia de comparación que facilite la utilización de los datos almacenados para disminuir las tasas de error obtenidas.

Referencias bibliográficas

1- **Maio, Dario, y otros**.Handbook of Fingerprint Recognition. Londres: Springer, 2009. ISBN 978-1-84882-253-5.

2- **Hernández León, Rolando Alfredo y Coello González, Sayda**.El proceso de investigación científica. Ciudad de La Habana: Editorial Universitaria del Ministerio de Educación Superior, 2011. ISBN 978-959-16-1307-3.

3- **Cappelli, Raffaele, y otros**. Fingerprint Image Reconstruction from Standard Templates. s.l. : IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, 2007.

4- **K. Jain, Anil, Nandakumar, Karthik y Nagar, Abhishek**. Fingerprint Template Protection: From Theory. 2012.

5- **Murillo-Escobar, M.A, Cruz-Hernández, C, Abundiz-Pérez, F, López-Gutiérrez, R.M** .Cifrado caótico de plantilla de huella dactilar en sistemas biométricos, 2014)

6- **K. Jain , Anil, Nandakumar, Karthik y Nagar, Abhishek**. Biometric Template Security. s.l. : EURASIP Journal, 2008.

7- **Muñoz, Alicia Hortensia Beisner**. ATAQUES TIPO “SIDE-CHANNEL" A SISTEMAS BIOMÉTRICOS DE RECONOCIMIENTO DE HUELLA DACTILAR. Madrid : Escuela politécnica superior, 2010.

8- **Duró, V. E.** (2009). Evaluación de Sistemas de Reconocimiento Biométrico, 58–61.

9- **Cruz, R.** (2009). Clasificación de huellas digitales mediante minucias.

10- **Ribalta, A. S.** (2012). Seguridad en los sistemas biométricos.

11- **K. Jain, Anil, Nandakumar, Karthik y Nagar, Abhishek**. Fingerprint Template Protection: From Theory. 2012.

12- **Nalini K. Ratha, Sharat Chikkerur, Jonathan H. Connell, Ruud M. Bolle**. Generating Cancelable Fingerprint Templates,2007

13- **K. Jain, Matoni, Maio, Prabhakar**. Handbook of Fingerprint Recognition, 2009.

14- **Fernández, R. S., Sentí, V. E., & Heredia, Y. H.** (2016). Cryptographic schemes for minutiae template protection, 14(4), 997–1004.

15- **Belguechi, Rima, Rosenberger, Christophe y Ait Aoudia, Samy.** BioHashing for securing fingerprint minutiae templates. s.l. : International Conference on Pattern Recognition, 2010.

16- **Arakala, A., & Sciences, G**. (2006). Jason Jeffers and Arathi Arakala. Biometrics.

17- **Favre, Mélanie, Bringer, Julien y Chabanne, Hervé**. FUZZY VAULT FOR MULTIPLE USERS Ifrane, Morocco : Sponsored by French ANR project BMOS, 2012.

18- **Ratha, N., Connell, J., & Bolle, R. M.** (2006). Cancelable Biometrics : A Case Study in Fingerprints. Pattern Recognition, 18–21.

19- **Pró, M. L., Juan, M., Gonzáles, C., Contreras, L. W., & Yañez, L. C.** (2012). Tecnologías Biométricas aplicadas a la seguridad en las organizaciones, 55–66.

20- **Wilson, Leslie Blacket**.Comparative Programming Languages . New York: Addison-Wesley, 1993. ISBN 0-201-56885-3.

21- **Lorente, Carlos del Junco**.Manual de Programacion C++. España: Didact S.L: Mad, 2005.

22- **Guéri, Brice-Arnaud**.Lenguaje C++. s.l.: ENI, 2005.

23- **Groussard, Thierry**.C# 4 Language Bases. Barcelona: ENI, 2011

24- **MARTÍNEZ, Layla Hirsh**. Intérprete y entorno de desarrollo para el aprendizaje de lenguajes de programación estructurada. 2007.

25- **Jornadas, V.** (2003). Metodologías Ágiles en el Desarrollo de Software.

26- **LÓPEZ, P.** Ingeniería del software I. Práctica 1. Herramienta CASE Visual Paradigm. 2012.

27- **Meza, José María Dueñas.** Lenguajes de programación. 2013.

28- **José H. Canós, Patricio Letelier y Mª Carmen Penadés.** Métodologías Ágiles en el Desarrollo de Software. DSIC -Universidad Politécnica de Valencia, Camino de Vera : s.n. 46022 Valencia.

29- **Extreme Programing, Agile Process.** Agile Software Development: A gentle introduction. [En línea] 2009. http://www.agile-process.org.

30- **Reynoso, Carlos y Kicillof, Nicolás.** Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. UNIVERSIDAD DE BUENOS AIRES : s.n., 2004.

31- **Pressman, Roger S.** Software Engineering: A Practitioner's Approach. 2010. 978-0-07-337-597-7.

32- **Camacho, Erika, Cardeso, Fabio y Nuñez, Gabriel.** Arquitecturas de Software. Abril

33- **Pressman, Roger S,Ph, D**. Ingeniería del software.2003 .

34- **Cooper, James W.** Java™ Design Patterns: A Tutorial. [ed.] Addison Wesley. January 28, 2000. pág. 352. ISBN: 0-201-48539-7.

35- **Mora, Roberto Canales.** Patrones de GRASP. Madrid : s.n., 2003-2004

36- Beck, K. Extreme Programming Explained. Embrace Change. Pearson Education. [trad.] Addison Wesley. 2008.

37- W. Cunningham and K. Beck, “tarjetas CRC.”

38- V. Jornadas, “Metodologías Ágiles en el Desarrollo de Software,” 2003.

39- Jeffers, Jason,Arakala, Arathi. 2007. FINGERPRINT ALIGNMENT FOR A MINUTIAE-BASED FUZZY VAULT. 2007.

40- Peña, Dr. Juan Manuel Fernández. Pruebas de software: experiencias en su aplicación. Capítulo 6 Pruebas de sistema. México : Facultad de Estadística e Informática Universidad Veracruzana, 2006.

41- Ingeniría del software. Un enfoque práctico. Pressman, Roger. Vol. 6ta edición.

42- Pressman, Roger S. Software Engineering: A Practitioner's Approach. 2010. 978-0-07-337-597-7.

43- Letelier, Patricio. Pruebas del Software. Universidad Politécnica de Valencia : Departamento Sistemas Informáticos y Computación.2009.

44- **K Schwaber, M Beedle. 2002.** Agile software development with Scrum. 2002.

45-**Arathi. 2007.** FINGERPRINT ALIGNMENT FOR A MINUTIAE-BASED FUZZY VAULT. 2007.

46- **J Palacio, C Ruata. 2010.** Scrum Manager. 2010.

47- Molpeceres, A. 2002. Procesos de desarrollo: RUP, XP y FDD. 2002.

48-OT Gómez, PPR López. 2014. Criterios de selección de metodologías de desarrollo de software. 2014.

49-RG Figueroa, CJ Solís, AA Cabrera. 2008. Metodologías tradicionales vs. Metodologías ágiles. 2008.

50- Y Araujo, H Lopez, A Mendoza, L Torrealba, G Ortiz. 2011. Metodología RUP (Rational Unified Process). 2011.

51-Bolle, N. K. Ratha J. H. Connell R. M. 2001. Enhancing security and privacy in biometrics-based authentication systems. 2001.

52-Fernández, Ramón Santana. 2016. Modelo para la protección de plantillas de minucias de huellas dactilares . 2016.

53- A. K. Jain, K. Nandakumar, and A. Nagar, “Fingerprint Template Protection : From Theory to Practice,” Secur. Priv. Biometrics, pp. 187– 214, 2013.

Anexos

Anexo 1

Historias de usuario

Historia de Usuario	
Número: HU_1	**Usuario:** Sistema
Nombre historia: Cargar la plantilla de minucias.	
Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Laura Elena González Varela	
Descripción: El sistema solicita la dirección del fichero a cargar, la cual es especificada por el usuario.	
Observaciones: -	

Tabla 10. "Historia de usuario correspondiente a la funcionalidad Cargar la plantilla de minucias"

Historia de Usuario	
Número: HU_2	**Usuario:** Sistema
Nombre historia: Leer la plantilla de minucias.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1

Programador responsable: Laura Elena González Varela.
Descripción: El sistema realizará todas las lecturas de las características de la plantilla de minucias cargada almacenando sus características es una lista de minucias.
Observaciones: -

Tabla 11. "Historia de usuario correspondiente a la funcionalidad Leer plantilla de minucias"

Historia de Usuario	
Número: HU_3	**Usuario:** Sistema
Nombre historia: Cifrar la plantilla de minucias mediante la transformación cartesiana.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Laura Elena González Varela.	
Descripción: El sistema realizará el cifrado de la plantilla de minucias, cambiando las celdas de tamaño fijo a una nueva posición especificada por la clave generada aleatoriamente.	
Observaciones: Cada minucia será ubicada en un sistema de coordenadas cartesiana, teniendo como referencia el núcleo y alineando el eje de las abscisas con su orientación.	

Tabla 12. "Historia de usuario correspondiente a la funcionalidad Cifrar la plantilla de minucias mediante la transformación cartesiana.".

Historia de Usuario	
Número: HU_5	**Usuario:** Sistema

Nombre historia: Almacenar la plantilla cifrada y la clave.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Laura Elena González Varela.	
Descripción: El sistema solicitará la dirección donde se desea almacenar la plantilla cifrada y los guardará en esta.	
Observaciones: La plantilla cifrada se almacenará en un fichero con extensión .pctc .	

Tabla 13. "Historia de usuario correspondiente a la funcionalidad Almacenar plantilla cifrada y la clave."

Historia de Usuario	
Número: HU_2	**Usuario:** Sistema
Nombre historia: Generar tripletas de minucias.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Laura Elena González Varela.	
Descripción: El sistema realizará todas las combinaciones posibles atendiendo a la cantidad de minucias en la plantilla y tomando tres de ellas en cada caso para la construcción de estructuras triangulares.	
Observaciones: Las estructuras triangulares a formar tendrán un conjunto de restricciones para probar que es un triángulo válido.	

Tabla 14. "Historia de usuario correspondiente a la funcionalidad Generar tripletas de minucias."

Historia de Usuario	
Número: HU_3	**Usuario:** Sistema
Nombre historia: Determinar las características identificativas de las tripletas de minucias.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Laura Elena González Varela.	
Descripción: Cada tripleta contendrá la longitud del desplazamiento de un punto a otro representando así los lados del triángulo, el ángulo relativo de cada minucia al lado correspondiente, el baricentro del triángulo y una estructura vector que contiene la longitud del desplazamiento del baricentro al vértice opuesto al lado más corto y su ángulo con respecto al eje x.	
Observaciones: -	

Tabla 15. "Historia de usuario correspondiente a la funcionalidad Determinar las características identificativas de las tripletas de minucias."

Historia de Usuario	
Número: HU_6	**Usuario:** Sistema
Nombre historia: Alinear la plantilla de minucias.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 4	**Iteración asignada:** 2
Programador responsable: Laura Elena González Varela.	

Descripción: Se alinean, aplicando rotación y traslación, al conjunto de puntos que representan las minucias de consulta para hacerlos corresponder.

Tabla 16. "Historia de usuario correspondiente a la funcionalidad Alinear la plantilla de minucias."

Historia de Usuario	
Número: HU_7	**Usuario:** Sistema
Nombre historia: Comparar las plantillas de minucias en el dominio cifrado.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 4	**Iteración asignada:** 2
Programador responsable: Laura Elena González Varela.	
Descripción: El sistema intentará encontrar al menos una coincidencia entre las estructuras triangulares generadas de los datos de consulta y las estructuras triangulares de la plantilla cifrada. . De lo contrario, en caso de no encontrarse coincidencia se asume que la plantilla de minucias de consulta no corresponde al rasgo biométrico protegido.	
Observaciones: -	

Tabla 17. "Historia de usuario correspondiente a la funcionalidad Comparar las plantillas de minucias en el dominio cifrado."

Anexo 2

Tarjetas CTR

Componente: Cifrado

Nombre de la clase: Cifrado	
Responsabilidades	**Colaboradores**

Cifrar mediante transformación cartesiana	1. Minucia
	2. Plantilla
Cifrar mediante transformación polar	1. Minucia
	2. Plantilla

Tabla 18. "Tarjeta CTR correspondiente a la clase Cifrado"

Componente Alineación

Nombre de la clase: Estructuracion	
Responsabilidades	**Colaboradores**
Combinaciones	-
Constructor	1. Minucia
	2. Triangulo

Tabla 19. "Tarjeta CTR correspondiente a la clase Estructuración"

Componente: Comparación

Nombre de la clase: Comparacion	
Responsabilidades	**Colaboradores**
ReordenacionCiclicaEspecial	1. Triangulo
CoincidenciaEstructural	1. Triangulo
BuscarCoincidenciaEstructural	1. Alineacion
VerificarRedundancia	-

Tabla 20. "Tarjeta CTR correspondiente a la clase Comparacion"

Componente: Visual

Nombre de la clase: Mainwindow	
Responsabilidades	**Colaboradores**
LeerPlantilla	2. Minucia
CifrarPlantilla	2. Minucia
	3. Triangulo

Tabla 21. "Tarjeta CTR correspondiente a la clase MainWindow"

Componente: Utililes

Nombre de la clase: Triangulo	
Responsabilidades	**Colaboradores**
CalcularDistancia	-
Baricentro	-
CalcularVModulo	-
CalcularVOrientacion	-
CalcularAngulo	-
CalcularAnguloRelativo	-
Reordenamiento	1. Minucia
Apto	-

Tabla 22. "Tarjeta CTR correspondiente a la clase Triangulo"

Nombre de la clase: Minucia	
Responsabilidades	**Colaboradores**
Constructor	1. Minucia

Tabla 23. "Tarjeta CTR correspondiente a la clase Minucia"

Anexo 3

Tareas de ingeniería

Iteración 2	
Historias de Usuario	**Tareas**
Alinear la plantilla de minucias.	1. Selección de la plantilla de consulta. 2. Selección del fichero los datos pertenecientes a la plantilla cifrada. 3. Búsqueda de coincidencias entre las estructuras triangulares. 4. Alineación de la plantilla de consulta, aplicando rotación y traslación con respecto al núcleo de la huella almacenada y rotando la plantilla de acuerdo a la diferencia angular entre los vectores de las estructuras triangulares coincidentes.
Comparar las plantillas en el dominio cifrado.	1. Compara la plantilla cifrada y la plantilla de consultas verificando su pertenencia al mismo rango biométrico.

Tabla 24. "Tareas de ingeniería Iteración 2"

Anexo 4

Imágenes de la aplicación

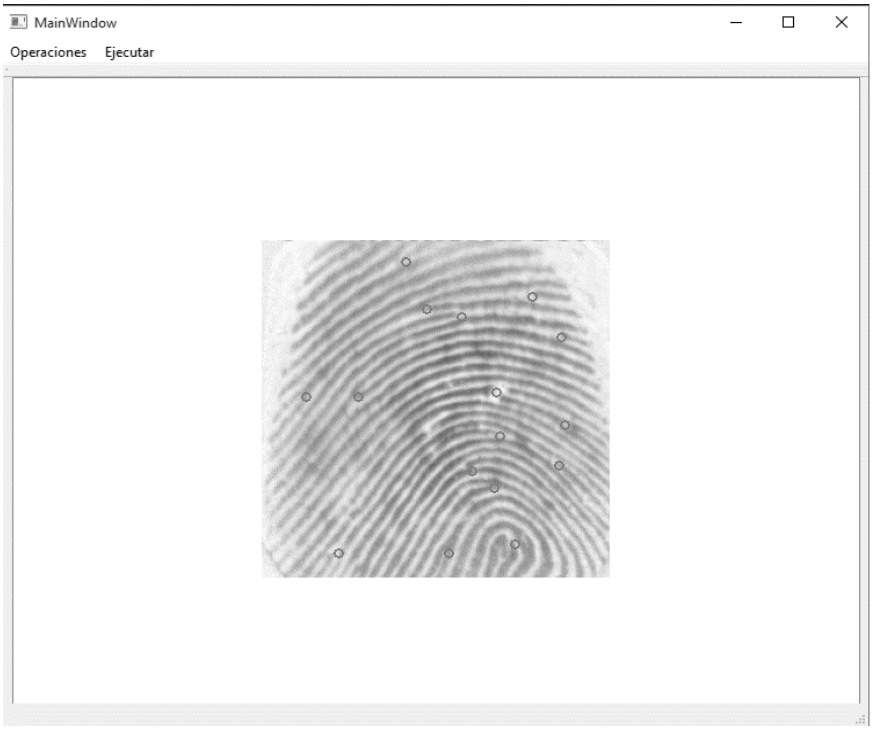


Figura 19. "Funcionalidad Cargar minucias"

Figura 20. "Formación de las tripletas de minucias"

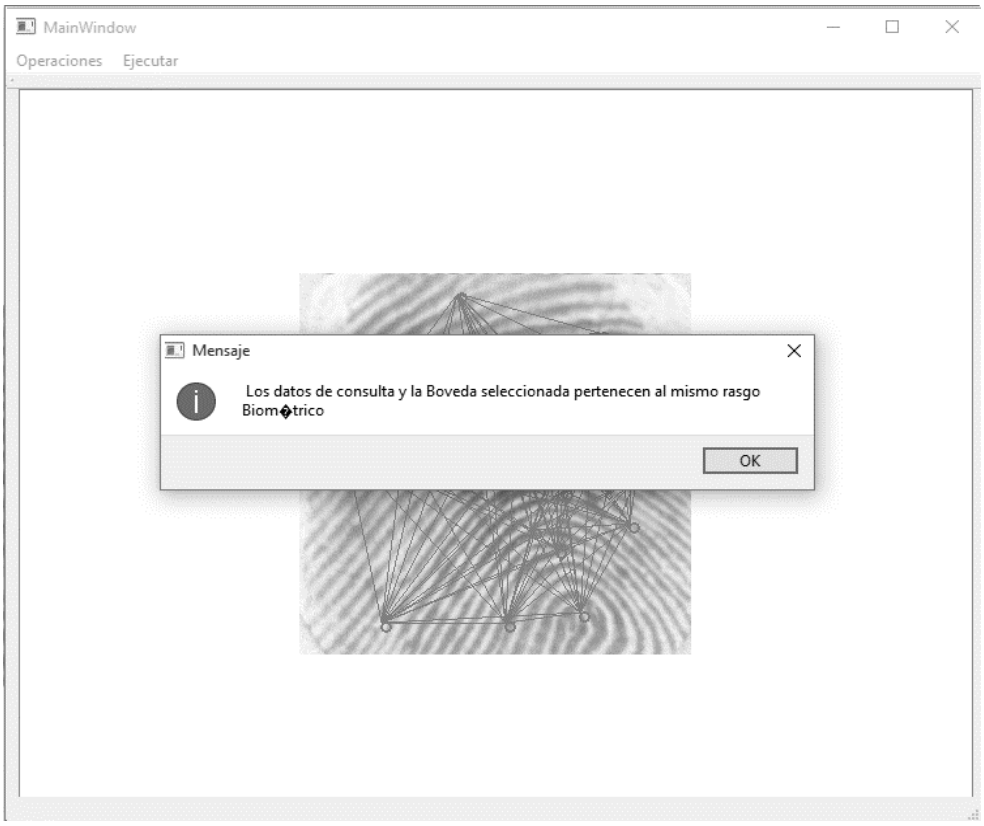


Figura 21. "Respuesta del sistema ante una comparación exitosa"