Universidad de las Ciencias Informáticas

Facultad 1

"Componente para la protección de plantillas de minucias de huellas dactilares utilizando bóveda difusa"

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

Autores: Dailín Martínez Pardo
Dashiel Hondarez Laza

Tutores: Ing. Ramón Santana Fernández
Ing. Yunierki Verdecia Viltres

La Habana, Junio 2015
"Año 57 de la Revolución"

"La diferencia entre lo que hacemos y lo que somos capaces de hacer, bastaría para solucionar la mayoría de los problemas del mundo"

Mahatma Gandhi (1869-1948)

DECLARACIÓN DE AUTORÍA

Declaramos ser los autores de la presente tesis que lleva por título: Componente para la protección de plantillas de minucias de huellas dactilares utilizando bóveda difusa y conferimos a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste se firma la presente a los ___ días del mes de junio del año 2015.

Dailín Martínez Pardo
Autor

Dashiel Hondarez Laza
Autor

Ing. Ramón Santana Fernández
Tutor

Ing. Yunierki Verdecia Viltres
Tutor

Datos del contacto

Tutor: Ing. Ramón Santana Fernández

Graduado de Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI) en el año 2011. Actualmente se encuentra realizando su investigación doctoral. Cuenta con 4 años de experiencia.

Correo electrónico: rsfernandez@uci.cu

Tutor: Ing. Yunierki Verdecia Viltres

Graduado de Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI). Actualmente se desempeña como profesor de Práctica Profesional en el Centro de Identificación y Seguridad Digital (CISED).

Correo electrónico: yvviltres@uci.cu

AGRADECIMIENTOS

A mi mamá por su amor incondicional, por tantos sacrificios, por estar ahí para mí y oírme sin nunca reprocharme nada, por entenderme siempre y perdonarme cuando me equivoqué, por ser la mejor mamá del mundo.

A mi papá, principal responsable de ser como soy, por enseñarme a ser cada día mejor, por sus maravillosos consejos, por confiar en mí y por ser un ejemplo de superación y dedicación.

A mi familia, por todo el apoyo que me han brindado siempre. A mi abuelo Eladio por ser un ejemplo de ser humano y por todo el cariño que me ha ofrecido.

A todos mis amigos y compañeros que me han apoyado y ayudado.

A Indy, Kilmer, Mary y Dashiel, quienes más que amigos son mis hermanos para toda la vida, por estar siempre a mi lado y apoyarme cuando más lo he necesitado.

A mi novio Osvi, por ser mi amigo, mi compañero y acercarme día a día a las estrellas.

Dailín Martínez

A mi mamá por su apoyo incondicional durante estos 23 años, por tanto amor, por sus regaños, por sus halagos, por ser ejemplo de dedicación y sacrificio, por darme todo cuanto pudo y a veces lo que no, en fin, por hacer de mí el hombre que hoy soy. No podrías ser mejor madre. Gracias.

A mi papá por enseñarme a pensar, por su guía insuperable en cada intento pero entender el mundo, por sus respuestas siempre acertadas a todas mis preguntas, por su amor, por sus regaños, por su carácter, por ser ejemplo de modestia y humildad, por ser el mejor papá del mundo.

A mi hermana Oli por ser la mejor hermana que se pueda tener, por aguantarme mis resabios como nadie más, por su cariño, por obligarme, inconscientemente, a ser ejemplo en todo, por ser mi niña por siempre.

A mi familia por todo el apoyo durante estos 5 años, por creer siempre en mí incluso cuando yo mismo dudaba. A mis abuelos, los que están y los que no, por una familia maravillosa.

A mis amigos de siempre por estar pendientes de cada paso que doy, por su preocupación y por su apoyo.

A Kilmer por ser mi hermano varón, a Michel a Liliana, a Ively por ser los amigos que supieron crucificarme cuando lo merecía y apoyarme cuando lo necesité.

A Dailín mi amiga y mi compañera de academia eterna, por los malos y buenos ratos que pasamos juntos. Gracias.

Dashiel Hondarez

DEDICATORIA

A la memoria de mis abuelas y mi abuelo Macho, quienes sé, estarían muy orgullosos de mí.

A mi mamita, quien es mi más grande tesoro y la persona más importante en mi vida.

A mi papá por ser como es y enseñarme tanto.

Dailín

A mi mamá, mi papá, mis hermanas y mis profes.

A todos los que me han enseñado de una forma u otra.

Dashiel

RESUMEN

En la actualidad son cada vez más utilizados los sistemas biométricos para proteger los recursos de forma efectiva. Los sistemas basados en huellas dactilares son los más utilizados e implementados, aunque no están exentos de posibles ataques. En un principio los puntos característicos que eran extraídos de la huella se almacenaban en texto plano, 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. Como solución a este problema se decide que una vez capturada la huella y extraídos estos puntos, se cifren para posteriormente ser almacenados. Aunque ya no se encuentran en texto plano, cuando va a realizarse la verificación biométrica, la plantilla cifrada debe descifrarse con el objetivo de ser emparejada con las características de consulta, por lo que los datos van a estar expuestos durante cada intento de autenticación. Esto trae como consecuencia que pueda ocurrir el robo o modificación no autorizada de la información, no garantizándose así la seguridad con esta técnica. La presente investigación tiene como finalidad desarrollar un componente que cifre los datos y efectúe la comparación directamente en el dominio cifrado y de esta forma mitigar la brecha de seguridad existente. Para lograr esto se lleva a cabo una valoración crítica de los diferentes modelos que se han planteado, a fin de encontrar la mejor opción para obtener la solución deseada. Se seleccionan las herramientas necesarias para la realización del componente, el cual se diseña, implementa y valida 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

La creciente evolución de las tecnologías de la informática y las comunicaciones demanda mejoras en los procesos de autenticación de personas. Las técnicas de identificación por conocimiento[1] y por posesión[2] han sido utilizadas durante años para autenticar a una persona en sistemas informáticos. Identificadores como la contraseña, las tarjetas inteligentes o los números de identificación personal (PIN) son algunos de los más utilizados, sin embargo la posibilidad de que sean adivinados, extraviados o robados disminuye sus niveles de seguridad.

La biometría es la ciencia que estudia las características físicas y conductuales de una persona que puedan ser de utilidad para identificarla (1). Entre las características más empleadas para el reconocimiento biométrico se encuentran la huella dactilar, la forma de la cara, la geometría de las manos, el iris y la voz (1). Las huellas dactilares en especial, presentan un conjunto de características como, gran facilidad de adquisición, son invariantes, inmutables y diversiformes (1), por lo que son utilizadas en casi el 50% de los sistemas para el reconocimiento de personas en la actualidad (2). Las características biométricas presentan baja probabilidad de ser robadas o transferidas.

Un sistema biométrico (3) es básicamente un sistema de reconocimiento de patrones que realiza la adquisición de los datos biométricos, extrae las características identificativas y realiza la comparación de estos con los datos almacenados previamente. En dependencia del contexto estos sistemas pueden realizar verificación o identificación de personas. La verificación es el proceso de determinar si una persona es quien dice ser, también denominado identificación positiva y se realiza comparando la muestra obtenida con una muestra específica almacenada en la base de datos. La identificación es el proceso que se realiza para conocer la identidad de una persona, comparando su muestra biométrica con todas las de la base de datos y se denomina identificación negativa (3).

[1] **Modelo de autenticación por conocimiento:** consiste en decidir si un usuario es quien dice ser a partir de una prueba de conocimiento, que solo él pueda superar, esa prueba no es más que una contraseña que en principio es secreta y está compuesta por un código alfanumérico y en ocasiones solamente numérico.

[2] **Sistema de autenticación por posesión:** utiliza un pequeño dispositivo de *hardware* que los usuarios cargan consigo para autorizar el acceso a un servicio de red. El dispositivo puede ser en forma de una tarjeta inteligente o puede estar incorporado a un objeto utilizado comúnmente.

Estudios realizados por diferentes autores (4; 5; 6) muestran un conjunto de vulnerabilidades en la arquitectura general de un sistema biométrico mediante las cuales es posible realizar ataques y afectar su correcto funcionamiento. Entre los más empleados se encuentran los ataques de denegación de servicio y de intrusión. Los ataques de intrusión tienen como principal objetivo obtener acceso ilegítimo a los recursos protegidos biométricamente. La 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 de minucias son los puntos específicos donde pueden obtenerse las plantillas biométricas para realizar este ataque, como se muestra en la figura 1.

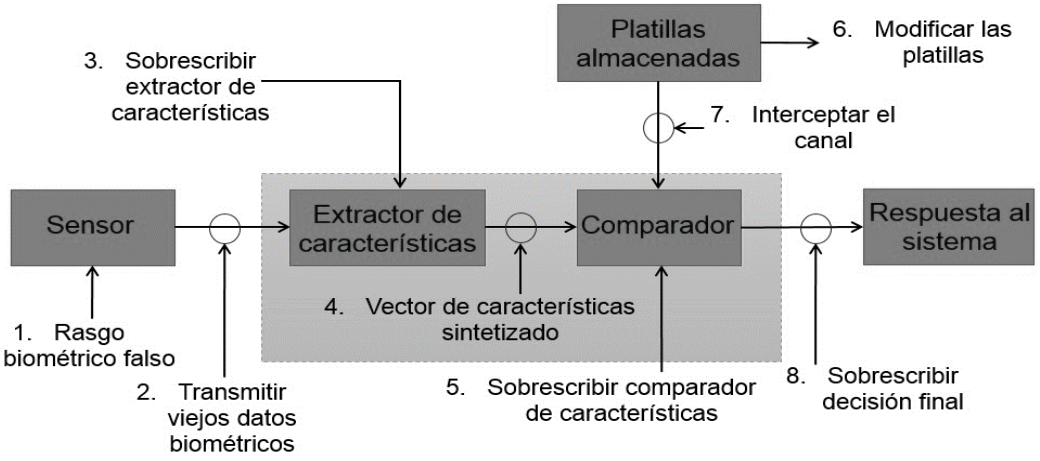

Figura 1. "Ataques a un sistema biométrico basado en huellas dactilares" (4)

Las plantillas de minucias, en el caso de un sistema automatizado de identificación mediante huellas dactilares, se transmiten y almacenan actualmente en texto plano. Para realizar la protección de estos datos se han utilizado métodos de criptografía tradicional (7) con el objetivo de disminuir la probabilidad de obtención de las minucias. Estos métodos utilizan funciones para el cifrado que son sensibles a variaciones. 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 el momento de la toma de la muestra, 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 van a estar expuestos durante cada intento de autenticación. El uso de algún virus del tipo Troyano[3] o un mal funcionamiento de *hardware* posibilitan la obtención de la plantilla de minucias en texto plano durante este proceso.

La obtención de la plantilla de minucias en texto plano trae como principales consecuencias que sea reconstruida la imagen de la huella dactilar como se describe en (8) para ser insertada en otros puntos del sistema biométrico como el módulo del sensor. La pérdida del identificador biométrico para toda la vida debido a que las huellas dactilares permanecen invariantes durante la vida de un individuo. La obtención de 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 proteger 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:** el proceso de protección de plantillas de minucias de huellas dactilares utilizando bóveda difusa.

Como **objetivo general de la investigación:** Desarrollar un componente que cifre los datos biométricos utilizando métodos criptográficos y códigos de corrección de errores para disminuir las fugas de información en plantillas de minucias de huellas dactilares.

Como **objetivos específicos:**

- Construir el marco teórico de la investigación a través de la extracción y recopilación de información asociada al problema de la investigación.
- Realizar el análisis y diseño de la propuesta de solución a partir del proceso de desarrollo de *software* utilizado para guiar la implementación del componente.
- Implementar el método para la protección de plantillas de minucias de huellas dactilares a fin de proteger los datos biométricos almacenados en estas.
- Validar la solución desarrollada a partir de la ejecución de pruebas de *software* para corregir los errores que puedan existir.

[3]**Programa tipo Troyano:** código ejecutable que no es directamente la traducción original del programa, sino que ha sido añadido con posterioridad y que entra simulando ser el original.

Idea a defender:

La utilización de métodos criptográficos y códigos de corrección de errores para el cifrado de plantillas de minucias de huellas dactilares disminuye las fugas de información en el proceso de reconocimiento de personas.

Métodos científicos

Métodos teóricos:

-Histórico-lógico: se utiliza para realizar una revisión cronológica de los principales modelos de protección de plantillas de minucias de huellas dactilares haciendo énfasis en sus principales características.

-Analítico-sintético: se utiliza para analizar las investigaciones previas que se han desarrollado, descomponiéndolas en diversas partes y sintetizando la información contenida en las partes previamente estudiadas, posibilitando la sistematización del conocimiento.

Métodos empíricos:

-Observación: es utilizado en el diagnóstico del problema a analizar y es de gran utilidad en el diseño de la investigación. Posibilita estudiar los procesos de forma general, sin tener que adentrarse en sus especificaciones.

Justificación de la investigación:

El uso y almacenamiento de las plantillas de minucias de huellas dactilares en texto plano en los sistemas de identificación de personas mediante huellas dactilares facilita la obtención de los datos biométricos en ataques realizados, lo que ocasiona principalmente la pérdida de seguridad y confidencialidad de la información. El uso de las técnicas tradicionales para el cifrado de los datos no soluciona el problema debido a que al realizar la verificación biométrica la plantilla debe ser descifrada, quedando vulnerable en este instante.

La presente investigación va dirigida a la obtención de un componente que realice el cifrado de los datos biométricos y compare los datos cifrados en el dominio protegido. De esta forma se reducen las probabilidades de obtención de las características biométricas y se garantiza la confiabilidad, privacidad y seguridad de los datos, reduciéndose así las vulnerabilidades de los sistemas biométricos que utilizan la huella dactilar como método de reconocimiento.

El trabajo estará estructurado de la siguiente forma:

Capítulo 1. Fundamentación teórica: en este capítulo se presentan los elementos teóricos metodológicos que facilitan la comprensión del problema planteado, se realiza una descripción de los conceptos fundamentales relacionados con el dominio del problema y se definen las tecnologías, metodologías y herramientas a utilizar en el desarrollo de la solución.

Capítulo 2. Propuesta de solución: en este capítulo se realiza una descripción de la propuesta de solución, se identifican los requisitos funcionales y no funcionales y se propone la arquitectura del sistema haciendo énfasis en los patrones de diseño empleados.

Capítulo 3. Implementación y pruebas: en este capítulo se describen los aspectos relacionados con el desarrollo de la solución, los estándares de codificación empleados y las clases u operaciones implementadas y se presentan las pruebas realizadas al componente.

1 CAPÍTULO 1: Fundamentación teórica

1.1 Introducción

En este capítulo se realiza un análisis de los principales conceptos para facilitar un mejor entendimiento del problema planteado. Se estudian diferentes modelos de protección de plantillas de minucias, destacando sus principales ventajas y limitaciones. Se seleccionan además la metodología, tecnologías y herramientas de desarrollo a utilizar durante el proceso de implementación de la solución.

1.2 Conceptos asociados al dominio del problema

Sistema biométrico:

Un sistema biométrico es un sistema de reconocimiento de patrones que opera a partir de la adquisición de datos biométricos de un individuo; extrae un conjunto de características de los datos capturados y las compara con las almacenadas en la base de datos (3). Los datos biométricos son obtenidos a partir de la medición de rasgos físicos y conductuales pertenecientes a una persona (1). El rostro, el termograma facial, la huella dactilar, la geometría de la mano, el patrón venoso de las manos, el iris, la voz, la firma manuscrita, entre otros, son algunos de los rasgos utilizados para autenticar biométricamente a una persona como se muestra en la figura 2.

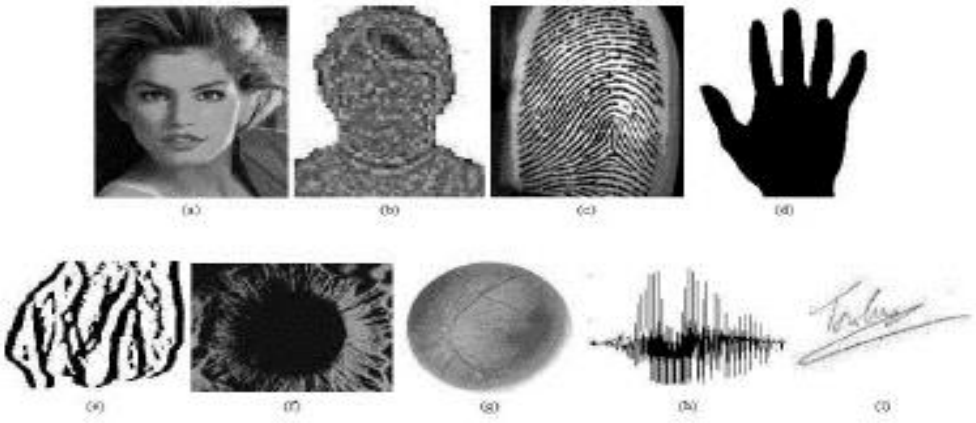

Figura 2. "Rasgos físicos y conductuales utilizados para el reconocimiento biométrico"

Reconocimiento biométrico:

El reconocimiento biométrico es definido en (1) como una manera de referirse indistintamente al proceso de verificación o identificación de personas.

- **Verificación de personas:**

Proceso mediante el cual se determina la identidad de una persona a partir de la comparación de la muestra biométrica de consulta con la almacenada en la base de datos (1). Esto es lo que se denomina "reconocimiento positivo" y requiere de una comparación "uno a uno". Con la puntuación de la salida de la comparación y a partir de un umbral, el sistema toma una decisión: se trata de un usuario genuino o un usuario impostor (3).

- **Identificación de personas:**

Proceso mediante el cual se determina la identidad de una persona a partir de la comparación de la muestra biométrica obtenida con todas las almacenadas en la base de datos (1). Se trata de una comparación "uno a muchos". El modo de identificación requiere de un coste computacional mucho más elevado que el modo de verificación. En este caso se trata de "reconocimiento negativo" y únicamente puede realizarse mediante sistemas biométricos (3).

Huella dactilar:

Las huellas dactilares se encuentran en la parte posterior de los dedos de las manos como reproducción de la epidermis. Están constituidas por rugosidades que forman salientes y depresiones. Los salientes se denominan crestas papilares y las depresiones surcos interpapilares (o valles). Una cresta está definida como un segmento de curva y un valle como la región entre dos crestas adyacentes (Ver Figura 3) (9).

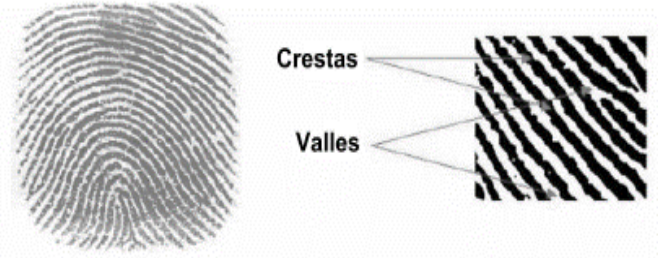

Figura 3. "Crestas y valles de una huella dactilar"

Propiedades de las huellas dactilares (1)**:**

- Perennes: desde que las huellas dactilares se forman en el sexto mes de la vida intrauterina, estas permanecen indefectiblemente invariables en número, situación, forma y dirección hasta que la putrefacción del cadáver destruye la piel.

- Inmutables: se evidencia cuando las crestas papilares no pueden modificarse fisiológicamente; si hay un traumatismo poco profundo, se regeneran y si es profundo, las crestas no reaparecen con forma distinta a la que tenían, sino que la parte afectada por el traumatismo resulta invadida por un dibujo de cicatriz.
- Diversiformes: no se han hallado todavía dos impresiones idénticas producidas por dedos diferentes.

Minucia:

Punto de interés de la huella dactilar, que se representa como $\{x, y, \theta, t\}$ donde x, y representan la posición en la imagen de la huella, θ es el ángulo en radianes de dicha minucia y t es el tipo de minucia (10).

Las minucias pueden ser clasificadas en diferentes tipos (ver figura 4), sin embargo de forma general se clasifican como terminaciones o bifurcaciones, pues el resto de las clasificaciones se ven como combinaciones de estas.

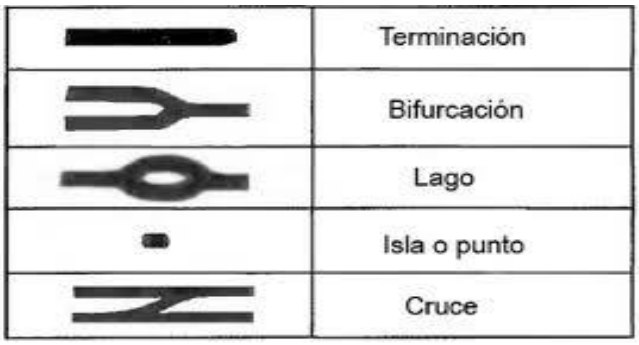

Figura 4. "Tipos de minucias" (11)

Protección de los datos biométricos:

Cuidado preventivo de los datos biométricos ante un intento de robo o violación en un sistema de reconocimiento de este tipo (12). Al proteger los datos biométricos se garantiza la integridad, confidencialidad y privacidad de estos, minimizando la posibilidad de acceso a ellos por usuarios no autorizados (6).

Para obtener plantillas biométricas seguras en (7; 4) se plantea que se deben cumplir los siguientes requisitos:

- Seguridad criptográfica: teniendo una plantilla segura, es computacionalmente más complicado encontrar un conjunto de características biométricas (comúnmente conocido como pre-imagen) que coincidan con la plantilla segura. Esta propiedad de pre-imagen reduce la posibilidad de que un atacante se introduzca en el sistema biométrico por concepto de reproducción de la pre-imagen.

- Rendimiento: el modelo de protección de plantillas de minucias no debe degradar el rendimiento del proceso de reconocimiento (*False Match Rate* (FMR) y *False Non-Match Rate* (FNMR) (1)) del sistema biométrico.
- Revocabilidad: es necesario tener un modelo de protección de plantillas que genere múltiples plantillas seguras del mismo dato biométrico. Esas plantillas seguras deben ser tales que aunque adversario obtenga dos o más de ellas, le sea computacionalmente difícil identificar que son derivadas del mismo dato biométrico y obtener la característica biométrica original del usuario. La propiedad de revocabilidad garantiza que pruebas cruzadas entre bases de datos biométricas no sean posibles, preservando así la privacidad del usuario. La revocabilidad también hace que sea fácil de descartar una plantilla comprometida y se vuelva a emitir una nueva basada en los mismos datos biométricos.

Idealmente los modelos de protección de plantillas deberían satisfacer estos tres requerimientos simultáneamente, sin embargo es todo un reto diseñar una técnica así (7).

1.3 Modelos de protección de plantillas de huellas dactilares

Los modelos de protección de plantillas de minucias de huellas dactilares se clasifican en basados en minucias o basados en patrones (7; 2). Los basados en minucias se dividen 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 (por ejemplo, distancias entre pares de 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. Debido a que los modelos basados en minucias son los más utilizados (1) se decide emplear un modelo de este tipo.

1.3.1 Plantillas cancelables

El modelo de protección de plantillas de minucias denominado plantillas cancelables o plantillas no invertibles (7; 11; 13) está compuesto por tres tipos de transformaciones aplicadas al conjunto de datos biométricos para su protección. Estas transformaciones son:

1. Transformaciones cartesianas
2. Transformaciones polares
3. Transformaciones funcionales

Transformación cartesiana: esta transformación se realiza en el plano de coordenadas cartesianas y comienza con la conversión de la plantilla de minucias en una rejilla rectangular, 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 específica del usuario.

Transformación polar: esta transformación se realiza de forma similar a la cartesiana aunque primeramente se lleva el plano de coordenadas cartesianas a polares, para posteriormente ser dividido en un determinado número de capas y cada una de estas dividida en sectores. Dado que el tamaño de los sectores puede ser diferente (cerca del centro son más pequeños que lejos del centro), se imponen restricciones a la traducción del vector generado a partir de la clave, como que la distancia radial del sector transformado no sea muy diferente de la distancia radial de la posición original.

Transformación funcional: 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 (14). 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.

Ejemplos de transformaciones cartesianas, polares y funcionales se ilustran en la figura 5.

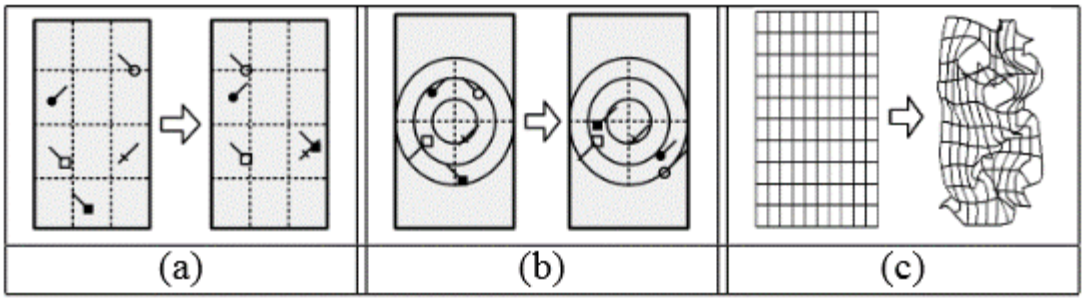

Figura 5. "Función de transformación cartesiana, polar y funcional para generar plantillas biométricas cancelables. (a) Minucias originales y transformadas a partir de la transformación cartesiana, (b) Minucias originales y transformadas después de la transformación polar, (c) Minucias originales y transformadas después de la transformación funcional" (14)

Estas funciones, utilizadas en el proceso de cifrado, son funciones de un solo sentido o no invertibles, las cuales garantizan la seguridad criptográfica de los datos. Este modelo brinda la oportunidad de cancelar la plantilla y su correspondiente transformación cuando los datos biométricos y/o las transformaciones se vean comprometidas. La característica principal que posee es que incluso si se conoce la clave y/o la plantilla

transformada, es computacionalmente difícil (en términos de complejidad de fuerza bruta) para un adversario recuperar la plantilla biométrica original (7).

Existen otros enfoques de este modelo (4; 15; 16; 17; 18) que resuelven un conjunto de limitaciones como la alineación, pero aún se evidencian deficiencias que no se han logrado mitigar, como:

- La disyuntiva entre discriminabilidad y no invertibilidad de la función de transformación. Esta función debe preservar la discriminabilidad (estructura de similitud) del conjunto de características. Tanto en el espacio original como en el transformado, las características del mismo usuario deben tener alta similitud y características de diferentes usuarios deben ser muy diferentes después de la transformación. La transformación debe ser no invertible, pues a partir de un conjunto de características transformadas debe ser difícil para un adversario obtener el conjunto de características originales (o una aproximación cercana de las mismas). Resulta complejo diseñar funciones de transformación que satisfagan tanto la discriminabilidad como condiciones de no invertibilidad simultáneamente.
- Dos o más minucias pueden ser asignadas a un mismo punto en el dominio transformado. En la transformación cartesiana, por ejemplo, dos o más celdas pueden ser mapeadas a una sola, de modo que no se puede determinar la celda original a la que pertenece un punto característico porque cada minucia puede pertenecer a alguna en específico independientemente de las celdas posibles y esto proporciona una cantidad limitada de no invertibilidad a la transformación.

1.3.2 ***BioHashing***

Este modelo de protección según el enfoque propuesto en (19) se divide en dos componentes: el primero es la transformación integral invariante y discriminativa de los datos de una huella dactilar, con un moderado grado de tolerancia de desvío y el segundo es la discretización de los datos mediante un producto interno de números aleatorios (tokens) y datos del usuario.

En el modelo *biohashing* se integra un conjunto de números al azar específicos del usuario con características biométricas para tratar el tema de la seguridad criptográfica y el rendimiento. Los vectores resultantes de la aplicación de esta técnica de protección son muy poco correlacionados debido a las variaciones que aparecen en el momento de la toma de la muestra. Es una forma no determinista para obtener el código específico del usuario sin tener ambos tokens con datos aleatorios y la función de la huella dactilar del usuario (19).

En (20) se presenta otro enfoque de este modelo, siendo los pasos a seguir:

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*.

Descripción del proceso:

Teniendo como entrada la plantilla de minucias extraída, 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[4]; si E>Tr (Tr es un umbral óptimo Otsu[5]) entonces S es válida. Posteriormente se filtra la región en ocho direcciones diferentes usando un banco de filtros de Gabor[6]. 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 $\forall f_{i\theta} \in \mathrm{F}, f_{i\theta} = \frac{1}{n_i} \sum_{n_i} | \ Im_{i\theta}(x,y) - p_{i\theta}|$, donde n_i es el número de píxeles en S_i y $p_{i\theta}$ es su media.

Para cada *minucode* mc establecido se realiza el proceso de creación del *biohashing* correspondiente de la siguiente manera:

1. Generar un conjunto de vectores aleatorios Γ.
2. Aplicar el proceso de Gram-Schmidt[7] para transformar el Γ base en un conjunto ortogonal de matrices $r_{\perp \mathbf{i}}$, 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 $r_{\perp \mathbf{i}}$ ($f \mid r_{\perp \mathbf{i}}$), donde $i_{(1..v)}$.

[4] **Espectro de Fourier:** espectro de frecuencias de una función.

[5] **Umbral óptimo Otsu:** método que permite calcular el mejor valor umbral automáticamente, utilizando la varianza como medida de dispersión de los niveles de gris.

[6] **Banco de filtros de Gabor:** funciones que pueden diseñarse como un banco de filtros con diferentes dilataciones y rotaciones.

[7] **Proceso de ortogonalización de Gram-Schmidt:** algoritmo para construir, a partir de un conjunto de vectores linealmente independientes de un espacio prehilbertiano, otro conjunto ortonormal de vectores que genere el mismo subespacio vectorial.

4. Calcular una cadena de bits de *biohashing* denotado b (b $\in \{0,1\}^{v}$),

   $b_{\mathbf{i}} = \begin{cases} 0\ si\ (f \mid r_{\perp \mathbf{i}}) \leq t \\ 1\ si\ (f \mid r_{\perp \mathbf{i}}) < t \end{cases}$, *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.

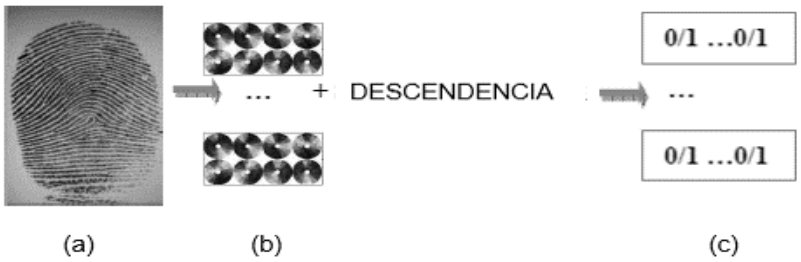

Figura 6. "Protección de plantillas de minucias por biohashing. (a) Extracción de minucias, (b) Minucodes, (c) Biocodes"

Para la comparación de dos vectores se realiza (20):

1. Correcta rotación.
2. Proceso de *biohashing* al conjunto de *minucodes* extraídos de la toma.
3. Llevar a cabo el algoritmo de coincidencia local (21) entre las dos plantillas correlacionadas.

Antes de realizar la comparación entre la plantilla de huellas dactilares almacenada y la plantilla de minucias de consulta se debe corregir la deformación de rotación existente entre ellas. La corrección consiste en ir rotando la imagen de entrada por la diferencia existente entre las dos orientaciones de referencia (22). En la figura 7 se corrige la rotación en comparación a la imagen (a) de la figura 6.

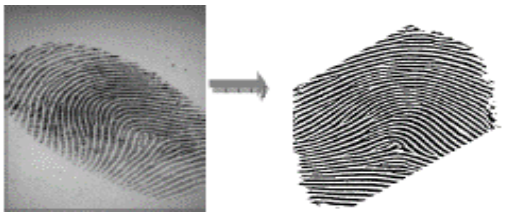

Figura 7. "Ejemplo de corrección de rotación"

Sea T = {t1,..., tm} y P = {p1,..., 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[8] 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 (20).

Algoritmo de comparación de tipo coincidente local:

Primeramente se define una vecindad local de minucias m_i $\{m_1, ..., m_k\}$ de k vecinos más cercanos de m_i en términos de Distancia Euclidiana[9].

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 (20).
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[10] entre *biocodes.* Finalmente se aplica la función de consolidación propuesta en (20):

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

Consideraciones a tener en cuenta para realizar el proceso de comparación en el dominio protegido utilizando el método descrito (20)

Realizar el proceso de comparación en el dominio protegido es un trabajo complejo que consta de consideraciones a tener en cuenta para la obtención de un correcto resultado. Algunas de estas consideraciones son:

- No todas las minucias se analizan pues depende del valor de B para que la región sea válida.

[8] **Variación intra-usuario:** dos impresiones del mismo dedo poseen diferente número y ubicación de las minucias debido a la traslación y rotación que se manifiesta en una impresión con respecto a la otra.

[9] **Distancia Euclidiana:** distancia "ordinaria" (que se mediría con una regla) entre dos puntos, la cual se deduce a partir del teorema de Pitágoras.

[10] **Distancia de Hamming:** número de bits que tienen que cambiarse para transformar una palabra de código válida en otra palabra de código válida.

- La elección de k es muy importante para evitar el problema de mínimo local.
- La fase 1 es sensible a falsas minucias.
- Siempre que se encuentre una función de consolidación igual a 0% significa que las plantillas son suficientemente distantes.

Limitaciones que presenta este modelo

Este modelo posee limitantes que dificultan su utilización, como son:

- Si se ve comprometida la clave específica del usuario la plantilla ya no es segura, porque la transformación normalmente es invertible, por lo que si un adversario obtiene acceso a la clave y a la plantilla transformada se puede recuperar la plantilla biométrica original (4).
- Dado que la comparación tiene lugar en el dominio transformado, el modelo *biohashing* necesita ser diseñado de tal manera que la capacidad de reconocimiento no se degrade, especialmente en la presencia de grandes variaciones intra-usuario (4).
- Es posible determinar el *biocode* a partir de un ataque de fuerza bruta (20).
- En este modelo la revocabilidad no es un proceso sencillo puesto que de aplicarla repercutiría en cambios significativos, lo que afectaría considerablemente el proceso de identificación (20).

1.3.3 **Bóveda difusa**

Es una construcción criptográfica que está diseñada para trabajar con los datos biométricos representados como un conjunto desordenado (por ejemplo las minucias en huellas dactilares) (11). Este término fue definido por primera vez en (23) en el cual se propone el algoritmo a utilizar para proteger la información de posibles intrusos. Este modelo funciona de la siguiente manera (7; 23):

Se toma el conjunto 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[11] 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

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

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.

Los tres parámetros principales que se plantean en la definición del modelo de bóveda difusa propuesto en (23) son r, q y k. El parámetro 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.

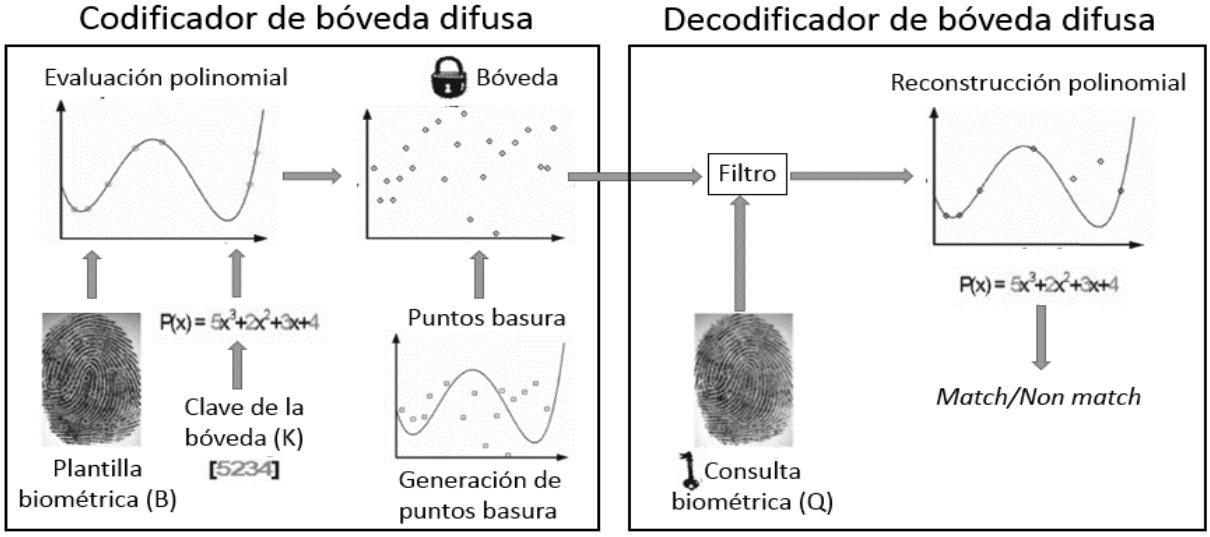

Figura 8. "Esquema de protección de plantillas de minucias usando bóveda difusa" (7)

La alineación es un requisito indispensable en cualquier modelo de protección, debido a los diferentes tipos de distorsión que pueden ocurrir durante la adquisición de los datos biométricos (24). El número de puntos de operación factible (en la bóveda se opera con una complejidad insignificante medida a través del número de intentos de acceso obligados a revelar el secreto de un usuario real y con una complejidad considerable para un usuario impostor) de la bóveda difusa es limitado.

T. C. Clancy, N. Kiyavash y D. J. Lin (2003) propusieron un modelo basado en la ubicación de las minucias (7) como modificación del modelo de bóveda difusa propuesto por Juels y Sudán (23). Ellos supusieron que

la plantilla y los conjuntos de minucias de consulta estaban pre-alineados, lo cual no es una hipótesis realista en sistemas de autenticación de huellas dactilares prácticos. Múltiples impresiones de huellas dactilares de un usuario fueron utilizadas durante la inscripción para la identificación de los puntos de minucias fiables. Se simuló la etapa de corrección de errores sin haber sido implementada en realidad. La tarifa de FNMR de su sistema fue de aproximadamente 20-30% y afirmó que recuperar el secreto era 269 veces más difícil para un atacante que para un usuario genuino (25).

Y. Dodis, L. Reyzin y A. Smith (2004) también plantearon una modificación del modelo de bóveda difusa de Juels y Sudán. En lugar de añadir puntos basura con las proyecciones del polinomio, propusieron utilizar un polinomio P` (de grado mayor que P) que se superpusiera con P solo para los puntos del conjunto SE. La seguridad del sistema se basa en el grado de este nuevo polinomio P`, que sustituye el punto final del conjunto yc (25).

S. Yang y I. Verbauwhede (2005) trataron de eliminar el requisito de pre-alineación del algoritmo propuesto por T. C. Clancy, N. Kiyavash y D. J. Lin (2003). La huella dactilar es analizada para encontrar una "minucia de referencia", la cual se define como una minucia que está presente en todas las impresiones de huellas dactilares múltiples (por ejemplo 3) y cuya estructura local (sobre la base de la distancia y orientación con respecto a sus dos vecinos más cercanos) no cambia.

La huella dactilar que bloqueará la bóveda se ha registrado con respecto a estas minucias de referencia (con la minucia de referencia como el origen de un nuevo sistema de coordenadas) y la bóveda se generará similar al algoritmo de T. C. Clancy, N. Kiyavash y D. J. Lin (2003), mediante el uso de minucias coordenadas (11).

Para el desbloqueo de la bóveda, los mismos pasos se aplican a las imágenes de huellas dactilares de consulta para encontrar a las otras "minucias de referencia". Si las dos minucias de referencia (la que se encuentra durante el bloqueo de la bóveda y la que se encuentra en el desbloqueo de la bóveda) son las mismas, el origen de los marcos de coordenadas utilizado en el bloqueo y desbloqueo sería el mismo y por lo tanto, la alineación se pudo establecer. Yang y Verbauwhede (2005) suponen que este sea el caso. Se debe tener en cuenta que este supuesto no garantiza que sea cierto, pero los autores no proporcionan detalles acerca de este tema (25).

En el 2009 otros autores (26) realizaron modificaciones al modelo de bóveda difusa propuesto por Juels y Sudán. Establecieron una libre alineación en el criptosistema de huellas dactilares fusionando las características locales, el descriptor de minucias y la estructura local de minucias, las cuales son invariantes

a la transformación en el proceso de captura de las huellas dactilares. Esta modificación propuesta posee como ventajas la posibilidad de evitar el procedimiento de alineación y mejorar el rendimiento y la seguridad del esquema de bóveda difusa al mismo tiempo, sin embargo conlleva a gastos más grandes en términos de almacenamiento de las plantillas.

En el 2012 en (27) se plantea otra modificación con respecto a la bóveda difusa clásica, la cual tiene como ventajas una decodificación única, un tamaño más pequeño de la bóveda para la misma entropía y necesita menos puntos basura debido a los múltiples conjuntos que se insertan en la bóveda. Este enfoque logra un esquema funcional y menos espacio de memoria, aunque posee limitaciones en cuanto al tamaño de la lista de salida, 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 crece rápidamente.

1.3.4 **Selección del modelo de protección a utilizar**

En el análisis realizado se encontraron los modelos bóveda difusa, plantillas cancelables y *biohashing*, de los cuales se considera que el que mejor realiza la protección de los datos es el de bóveda difusa debido a:

- En el modelo *biohashing* si se ve comprometida la clave específica del usuario la plantilla ya no es segura, porque la transformación normalmente es invertible. Si un adversario obtiene acceso a la clave y a la plantilla transformada puede recuperar la plantilla biométrica original, no pasando así en el modelo seleccionado pues la seguridad del sistema se basa en la inviabilidad del problema de la reconstrucción del polinomio.
- En el modelo basado en plantillas cancelables la función de transformación debe preservar la discriminabilidad (estructura de similitud) del conjunto de características. Tanto en el espacio original como en el transformado las características del mismo usuario tienen alta similitud y características de diferentes usuarios son muy diferentes después de la transformación. La transformación también debe ser no invertible, pues a partir de un conjunto de características transformadas debe ser difícil para un adversario obtener el conjunto de características originales (o una aproximación cercana de la misma). Es difícil diseñar funciones de transformación que satisfagan tanto la discriminabilidad como condiciones de no invertibilidad simultáneamente.
- En el modelo de bóveda difusa en lugar de proporcionarse una decisión "*Match/No-match*", la decodificación emite una clave que está embebida en la bóveda, que es a su vez la que se utiliza en el proceso de cifrado de la plantilla. Este modelo es tolerante a las variaciones intra-usuario en datos biométricos y esta tolerancia es determinada por el código de corrección de errores asociado a este.

1.4 Alineación de plantillas de minucias de huellas dactilares

La alineación es un requisito indispensable en cualquier modelo de protección debido a los diferentes tipos de distorsión (rotación, traslación, deformación no lineal y superposición parcial) que pueden ocurrir durante la adquisición de los datos biométricos (7). Aunque la alineación de dos huellas dactilares es un problema difícil en cualquier sistema de autenticación de personas, es mucho más complejo cuando la información sobre la plantilla no debe ser filtrada; constituyendo esto un problema de investigación activo.

Entre los modelos de alineación utilizados para resolver este problema se encuentra el empleo de las **estructuras de minucias** (28); las cuales son los rasgos que caracterizan a la información relativa entre dos o más minucias (por ejemplo, la distancia entre dos puntos característicos). 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 (28).

Otro modelo de alineación muy utilizado es la determinación de los **puntos singulares** (núcleo y delta) (1). Existen diferentes enfoques como el método del índice de Poincaré (29), 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 (1).

Otro modelo utilizado es el **punto focal de localización,** propuesto en (30), 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. Este algoritmo es iterativo y en cada iteración solamente se utiliza el campo de orientación en la región semi-circular de radio especificado con centro en el punto focal actual para generar los puntos de cruce. 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.

En (31) se propone otro modelo de alineación a utilizar denominado **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.

La determinación del **punto más cercano** es otro 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 (28). 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.

Los puntos de máxima curvatura tienden a ocurrir cerca de los puntos singulares de la imagen de la huella dactilar (32; 33). 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. Debido a que los puntos de alta curvatura son características globales del patrón de huella dactilar, no revelan ninguna información sobre los atributos de las minucias, que son las características locales de la huella dactilar.

1.4.1 Selección del modelo de alineación a utilizar

Entre los modelos de alineación encontrados durante la investigación realizada están las estructuras de minucias, los puntos singulares, el punto focal de localización, una minucia estable y el punto más cercano; de los cuales se considera que el que mejor cumple las necesidades es el uso de estructuras de minucias debido a:

- Las estructuras de minucias trabajan con un conjunto de características extraídas de las minucias que son invariantes a la rotación y traslación y resistentes a la deformación no lineal.
- Los puntos singulares, el punto focal de localización, una minucia estable y el punto más cercano son modelos muy sensibles a la obtención de las minucias, aspecto que es muy significativo.
- Los puntos singulares pueden o no aparecer.
- La necesidad de seleccionar cuidadosamente el punto focal de localización para la primera iteración es esencial, debido a su naturaleza iterativa.
- Es difícil determinar el punto característico estable de forma fiable.

Dentro de las estructuras de minucias más utilizadas (34) se encuentran las estructuras triangulares, la estructura de cinco vecinos más cercanos y la estructura de vecinos propuesta por Voronoi, siendo la más

eficiente el uso de estructuras triangulares con el objetivo de alinear plantillas. Se utilizará además la singularidad núcleo de la huella dactilar para alinear las plantillas como se propone en (34).

1.5 Metodologías de desarrollo de *software*

Algunos autores (35; 36) definen una metodología como una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de *software* en sus esfuerzos por implementar nuevos sistemas de información. Está formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo. En las dos últimas décadas, respecto a estas metodologías se ha establecido un intenso debate entre dos grandes corrientes. Las denominadas metodologías tradicionales, centradas en el control del proceso, con un riguroso seguimiento de las actividades involucradas en ellas y las metodologías ágiles, centradas en el factor humano, en la colaboración y participación del cliente en el proceso de desarrollo.

1.5.1 Metodologías tradicionales

Las metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo de *software*, con el fin de conseguir un *software* más eficiente. Para ello, se hace énfasis en la planificación total del trabajo a realizar y una vez que esté todo detallado, comienza el ciclo de desarrollo del producto. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada.

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

RUP constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. No es un sistema cerrado, más bien es un conjunto de metodologías adaptables al contexto y necesidades de cada organización. Su ciclo de vida es una implementación del Desarrollo en espiral[12]. Es un producto de Rational (IBM). Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso (37). Constituye una forma disciplinada de asignar tareas y responsabilidades, que pretende implementar las mejores prácticas en ingeniería de *software*, se basa en la administración de requisitos, control de cambios, modelado visual de *software* y verificación de la calidad de este.

[12]**Desarrollo en Espiral (Boehm 1988):** serie de ciclos que se repiten en forma de espiral comenzando desde el centro.

Divide el proceso de desarrollo en cuatro fases (38):

- Fase de inicio: abarca la comunicación con el cliente, las actividades de planeación y destaca el desarrollo y refinamiento de casos de uso como un modelo primario.
- Fase de elaboración: incluye la comunicación con el cliente y las actividades de modelado con un enfoque en la creación de modelos de análisis y diseño, con énfasis en las definiciones de clases y representaciones arquitectónicas.
- Fase de construcción: se refina y después se traduce el modelo de diseño en componentes de *software* ya implementados.
- Fase de transición: se transfiere el *software* del desarrollador al usuario final para realizar las pruebas betas y obtener la aceptación.

Estas fases se realizan durante varias iteraciones en número variable según el proyecto, haciendo un mayor o menor énfasis en distintas actividades durante las que se pueden distinguir nueve flujos o disciplinas a realizar en mayor o menor medida en dependencia de cada etapa. (Ver figura 9)

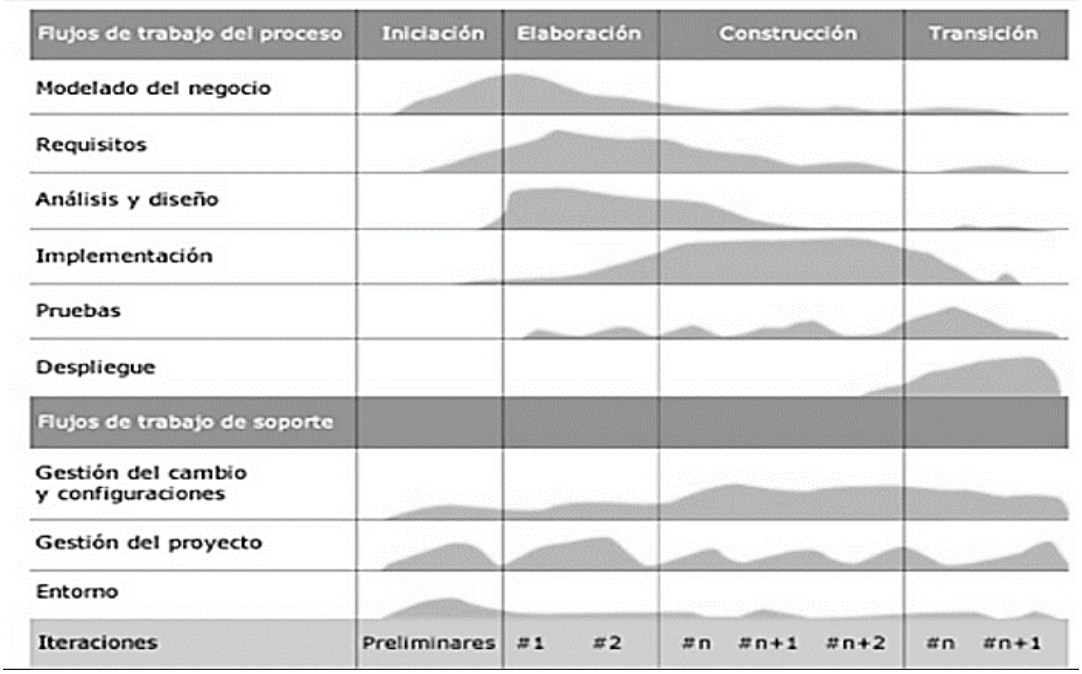

Figura 9. "Flujo de trabajo y fases de RUP" (36)

1.5.2 **Metodologías ágiles**

Los procesos de desarrollo rápido de *software* están diseñados para producir *software* útil de forma rápida. Generalmente, son procesos interactivos en los que se entrelazan la especificación, el diseño, el desarrollo y las pruebas.

XP

La programación extrema XP es posiblemente el método ágil más conocido y ampliamente utilizado. El nombre de XP fue acuñado por Beck (2000), debido a que el enfoque fue desarrollado utilizando las mejores prácticas del desarrollo iterativo y con la participación extrema del cliente. Aquí todos los requerimientos se expresan como escenarios (llamados historias de usuario), los cuales se implementan directamente como una serie de tareas. El desarrollo es de tipo incremental y se lleva a través de entregas pequeñas y frecuentes del sistema, por medio de un enfoque que sirve para la descripción de requerimientos basado en las historias de los clientes o escenarios que pueden ser la base para el proceso de planificación. En XP los clientes están implicados en la especificación y establecimiento de prioridades de los requerimientos del sistema (39). Dichos requerimientos no se especifican como una lista de funciones requeridas en el sistema. Los clientes del sistema son parte fundamental del equipo de desarrollo, esto permite que discutan escenarios con todos los miembros del equipo.

El ciclo de vida propuesto por la metodología se divide en las siguientes fases (40):

- Fase de exploración: 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.
- Fase de planificación: el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario, además de las entregas relacionadas con estas. El resultado de esta fase es un Plan de Entregas.
- Fase de iteraciones: esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entregas está compuesto por las diferentes iteraciones. En la primera iteración se intenta establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto, escogiendo las historias de usuario que fuercen la creación de esta arquitectura. No siempre esto es posible debido a que es el cliente el que determina las historias que se implementarán en cada iteración. Al final de la última iteración el producto estará listo para entrar en producción.

- Fase de puesta en producción: en esta fase se entregan módulos funcionales y sin errores y según los intereses del cliente el sistema puede ponerse en producción o no. No se realizan desarrollos funcionales, pero pueden ser necesarias tareas de ajuste.
- Fase de 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 esto se requiere de tareas de soporte para el cliente. La velocidad de desarrollo puede disminuir después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.
- Fase de muerte del proyecto: el cliente no tiene más historias para ser incluidas en el sistema, haciendo necesario que se satisfagan sus necesidades en otros aspectos tales como rendimiento y confiabilidad del mismo. Se genera la documentación final 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 no hay presupuesto para mantenerlo.

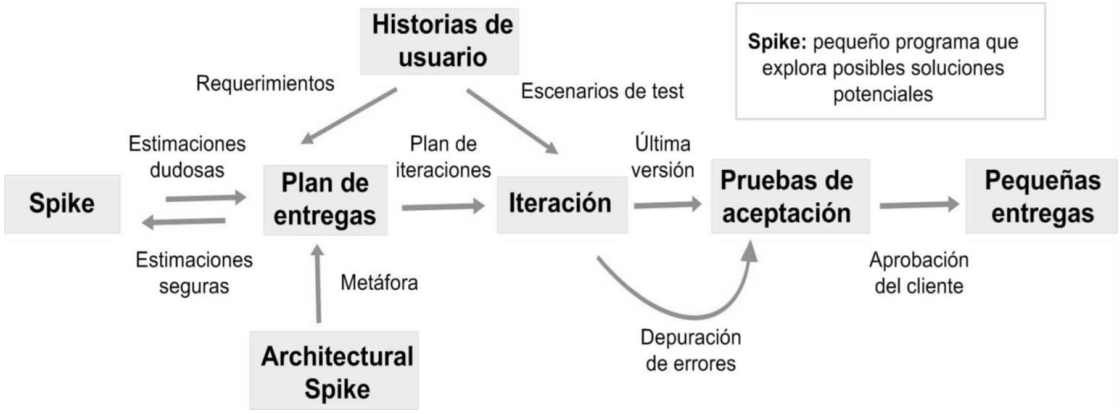

Figura 10. "Flujo de trabajo en XP" (40)

SCRUM

SCRUM (*Schwaber & Shuterland,* 2011*) (Shuterland,* 2012) aplica las mismas premisas conceptuales que XP pero para resolver un problema ligeramente distinto como es el de desarrollo evolutivo de aplicaciones. Es una metodología ágil y flexible que sirve para gestionar el desarrollo de *software*. Se basa principalmente en construir la funcionalidad de mayor valor para el cliente y en los principios de inspección continua, adaptación, autogestión e innovación.

Con SCRUM el cliente es pieza fundamental en el desarrollo de *software*, se entusiasma y se compromete con el proyecto dado que lo ve crecer iteración a iteración. Asimismo le permite en cualquier momento realinear el *software* con los objetivos de negocio de su empresa, pues puede introducir cambios funcionales o de prioridad en el inicio de cada nueva iteración. Esta forma de trabajo promueve la innovación, motivación y el compromiso del equipo que forma parte del proyecto, por lo que los profesionales encuentran un ámbito propicio para desarrollar sus capacidades. SCRUM genera algunas ventajas a diferencia de otras metodologías ágiles (39):

- 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.

- Reducción de riesgos: El hecho de llevar a cabo las funcionalidades de más valor en primer lugar y de conocer la velocidad con que el equipo avanza en el proyecto, permite despejar riesgos eficazmente de manera anticipada.

1.5.3 Selección de la metodología de desarrollo de *software* a utilizar

El análisis de diferentes metodologías de desarrollo de *software*, teniendo en cuenta las particularidades del componente a desarrollar, permitieron determinar la utilización de la metodología XP debido a:

- A pesar de que RUP busca constantemente calidad en sus procesos, llevando a cabo un robusto análisis, diseño, implementación, soporte y documentación de sistemas informáticos, contando para esto con un equipo de desarrollo grande y un completo dominio de todos los requisitos que cumplirá, no se ajusta a las particularidades de la solución a implementar. El grupo de trabajo es pequeño, lo cual dificulta la adopción de roles en cada etapa así como la generación de múltiples artefactos, indicando la necesidad de usar una metodología ágil.
- Aunque SCRUM es una metodología ágil se enfoca en las prácticas de organización y gestión, en vez de en la programación como XP. Es la indicada para proyectos con un rápido cambio de requisitos y su poca documentación puede provocar efectos negativos en el modo de confeccionar el producto final.
- Esta metodología ágil está caracterizada por un código simple y abierto a los cambios. Es apropiada para entornos volátiles y se define para equipos de desarrollo pequeños. El equipo de desarrollo definido tiene acceso todo el tiempo al cliente. Este debe permanecer disponible para responder preguntas y fijar prioridades, de forma que no atrase la toma de decisiones, facilitado así la retroalimentación y una planificación más transparente. Todo esto propicia que la tensión no esté solo en la entrega final del producto sino a lo largo de todo el proceso de desarrollo.

1.6 Tecnologías y herramientas

1.6.1 Lenguaje de modelado

El lenguaje de modelado en el proceso de desarrollo de *software* es esencial, repercutiendo en la etapa de diseño e implementación. Comunica la estructura del sistema a partir de un conjunto de notaciones, herramientas y prácticas, logrando un nivel de abstracción que organice la lógica del mismo y permita descubrir oportunidades de simplificación y reutilización.

UML

El Lenguaje Unificado de Modelado (LUM o UML por sus siglas en inglés, *Unified Modeling Language*) es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de los sistemas de *software* (41). En otras palabras UML se utiliza para definir un sistema de *software*. Posee la riqueza suficiente para crear un prototipo del sistema, pudiendo modelar, utilizando varios tipos diferentes de diagramas, los procesos de negocios, funciones, esquemas de bases de datos, expresiones de lenguajes de programación (42), entre otros aspectos.

Es fundamental aclarar que UML no es un lenguaje de programación, sino un lenguaje de modelado de propósito general, que ha demostrado su efectividad sobre todo en el área del análisis y diseño de sistemas de cómputo (43).

1.6.2 Herramientas de diseño

Una herramienta CASE (Ingeniería de *Software* Asistida por Computadoras, o por sus siglas en inglés, *Computer Aided Software Engineering*) es una aplicación informática destinada a aumentar la productividad en el desarrollo de *software*, reduciendo el coste de la misma en términos de tiempo y de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del *software*, en tareas como el diseño de proyectos, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores (44). Es un sistema de *software* que intenta proporcionar ayuda automatizada a las actividades del proceso de desarrollo.

Rational Rose Enterprise Edition

Es una herramienta CASE desarrollada por *Rational Corporation* basada en el Lenguaje Unificado de Modelación (UML), que permite crear los diagramas que se van generando durante el proceso de ingeniería en el desarrollo del *software*. Esta herramienta posibilita que el equipo de desarrollo entienda mejor el problema, que identifique las necesidades del cliente de forma más efectiva y comunique la solución propuesta de manera más clara (45). El desarrollo es un proceso iterativo, que comienza con una aproximación del análisis, diseño e implementación para identificar los riesgos y probar el sistema. Cuando la implementación pasa todas las pruebas que se determinan, se añaden los elementos modificados al modelo y una vez actualizado este, se realiza la siguiente iteración (46).

Las desventajas de esta herramienta residen en su enfoque fijo de desarrollo, los costos en *software*, manuales y capacitación y su limitación en la flexibilidad de la documentación. A pesar de esto, posee como

ventajas su utilidad en aplicaciones grandes y complejas. Con esta herramienta se reduce el tiempo de desarrollo de manera automática para pasar de un esquema a otro y al código, además de que los nombres y los datos se mantienen de manera consistente proporcionando una sincronización para diferentes desarrolladores.

Visual Paradigm para UML

Es una de las herramientas CASE más usadas, está compuesta por productos que facilitan a las organizaciones la visualización y diseño de diagramas. Sus soluciones se enfocan en eliminar la complejidad, aumentando así la productividad y disminuyendo el tiempo de desarrollo de las aplicaciones informáticas (47). Incorpora el soporte para trabajo en equipo, que permite que varios desarrolladores laboren a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros. Incorpora el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue.

El *software* de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite construir todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Visual Paradigm cumple con las políticas actuales de migración a *software* libre, siendo multiplataforma, de forma tal que facilita la modelación del *software* independientemente del sistema operativo que se emplee (48).

1.6.3 Lenguajes de programación

Los lenguajes de programación son herramientas que permiten crear programas y *software*. Facilitan la tarea de programación, pues disponen de formas adecuadas que pueden ser leídas y escritas por personas, a su vez resultan independientes del modelo de computador a utilizar (49). Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones (50).

C++

Bjarne Stroustrup crea la primera versión experimental de este lenguaje hacia 1979, con la intención de proporcionar una herramienta de desarrollo para el núcleo Unix en ambientes distribuidos. En 1983 el lenguaje se rebautiza 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. Desde sus inicios, C++ intentó ser un lenguaje que incluye completamente al lenguaje C, pero al mismo tiempo incorpora muchas características sofisticadas no incluidas en este (51), tales como programación orientada a objetos, excepciones, sobrecarga de operadores y plantillas.

C++ es un lenguaje de programación de propósito general que ofrece economía sintáctica, control de flujo, estructuras sencillas y un buen conjunto de operadores. No es un lenguaje de muy alto nivel, más bien es un lenguaje pequeño, sencillo y no está especializado en ningún tipo de aplicación. Esto lo hace un lenguaje potente, con un campo de aplicación ilimitado y sobre todo, se aprende rápidamente. En poco tiempo, un programador puede utilizar la totalidad del lenguaje (47). Desde el punto de vista de la Programación Orientada a Objetos (POO), C++ es un lenguaje híbrido. Además posibilita la redefinición de los operadores y crear nuevos tipos que se comporten como tipos fundamentales.

Java

Java es un lenguaje de programación portable sobre diferentes plataformas que fue desarrollado originalmente por la compañía Sun Microsystems (la cual fue adquirida por la compañía Oracle en 2010). Es un lenguaje de propósito general, orientado a objetos, basado en clases y concurrente, cuyo diseño está guiado a tener pocas dependencias de implementación y su sintaxis se deriva de C++ (52). Integra librerías estándares para interfaces de usuario, objetos distribuidos, hilos de ejecución, XML, web, móviles, tv, entre otras.

En realidad, Java hace referencia a un conjunto de tecnologías entre las cuales el lenguaje Java es solo una de ellas. Por tal motivo muchas veces se habla de la "plataforma Java", la cual es indesligable del lenguaje (51).

CSharp (C#)

C# es un lenguaje orientado a objetos elegante y con seguridad de tipos, que permite a los desarrolladores crear una amplia gama de aplicaciones sólidas. Está basado en las experiencias adquiridas de los lenguajes anteriores. Su sintaxis básica proviene de C y C++ y combina las mejores cualidades de un lenguaje de programación de alto nivel en la misma medida en que brinda un gran rendimiento (53). Es un lenguaje que posee muchas semejanzas con Java como son el estándar de código y el tipado seguro y unificado.

Como lenguaje orientado a objetos, C# admite los conceptos de encapsulación, herencia y polimorfismo. No hay archivos de encabezado independientes, ni se requiere que los métodos y los tipos se declaren en

un orden determinado. Un archivo de código fuente de C# puede definir cualquier número de clases, estructuras, interfaces y eventos (54).

1.6.4 Elección de las tecnologías y herramientas a utilizar

Las herramientas y tecnologías propuestas para el desarrollo de la solución son:

- UML para el modelado del *software*, permitiendo definir el sistema de *software*.
- Visual Paradigm para UML en su versión 8.0 pues es multiplataforma, permite que varios desarrolladores laboren a la vez en el mismo diagrama y vean en tiempo real los cambios producidos. Genera documentación del proyecto automáticamente en varios formatos e incorpora el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue.
- Se utilizará C++ como lenguaje de programación pues ofrece economía sintáctica, control de flujo, estructuras sencillas y un buen conjunto de operadores. Es un lenguaje potente, con un campo de aplicación ilimitado y sobre todo, se aprende rápidamente.

1.6.5 Plataforma de desarrollo

Para el desarrollo del componente se utilizará el *Framework* Qt en su versión 5.4.0. Qt es multiplataforma y un *framework* de interfaz de usuario utilizado para la creación de dispositivos y el desarrollo de aplicaciones de soporte a más de una docena de plataformas líderes. Posee librerías, clases y un entorno de desarrollo integrado: Qt Creator.

1.6.6 Entorno Integrado de Desarrollo

Un entorno de desarrollo integrado o IDE (*Integrated Development Environment*, por sus siglas en inglés) es un programa que incluye un conjunto de herramientas mediante las cuales los programadores pueden codificar (55).Entre las herramientas que componen un IDE están un editor de texto, un compilador, un intérprete, un depurador, un sistema de ayuda para la construcción de interfaces gráficas de usuario (GUI) y, opcionalmente, un sistema de control de versiones (56).

Se empleará 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. Es utilizado para desarrollar aplicaciones

con interfaz gráfica de usuario, así como para el desarrollo de programas sin interfaz gráfica, como herramientas para la línea de comandos y consolas para servidores (57).

1.7 Conclusiones parciales

El análisis de los conceptos fundamentales, en correspondencia con el tema tratado, proporcionó una mejor comprensión de los fundamentos teóricos de los modelos de protección de plantillas de minucias de huellas dactilares. El estudio de la bibliografía referente a los diferentes modelos permitió la selección de un conjunto de algoritmos para realizar la protección. La selección de la metodología, lenguajes, tecnologías y herramientas que se utilizarán en el desarrollo de la aplicación significó un paso importante para la obtención de un componente eficiente.

2 CAPÍTULO II: Propuesta de solución

2.1 Introducción

En este capítulo se abordan las principales características de la solución propuesta. Se define el modelo de dominio para comprender los principales conceptos con los que se trabaja. Se realiza el levantamiento de los requisitos funcionales y no funcionales, representando los funcionales en las historias de usuario. Se desarrollan las fases iniciales de la metodología de desarrollo XP, planificación y diseño, además de presentar los diferentes artefactos, los cuales serán premisas cruciales para la entrega final del sistema.

2.2 Propuesta de solución

El componente a desarrollar está basado en el modelo de bóveda difusa propuesto en (7) con algunas variaciones, como el tratamiento de la clave. Estará compuesto por la implementación de dos algoritmos: el primero realizará el proceso de cifrado de los datos para la construcción de la bóveda difusa y el segundo realizará el proceso de comparación de estos con respecto a uno de consulta.

Cifrado

Como primer paso para realizar el algoritmo de cifrado, una vez obtenida la plantilla de minucias, 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 (tripletas S_i). Posteriormente se extraerán de estas un conjunto de características. A continuación se evaluará cada minucia en el polinomio P establecido a partir de una clave k generada aleatoriamente, obteniéndose para cada una de ellas sus evaluaciones en el polinomio atendiendo a los valores de x y de y. Seguidamente se genera un conjunto de puntos basura, los cuales se mezclan aleatoriamente con los originales con el objetivo de protegerlos. Queda constituida así la bóveda difusa, a la cual se puede acceder, junto a un grupo de estructuras S_n, siendo n<=i, seleccionadas aleatoriamente. Además se almacena un archivo que contendrá la clave con la que fue cifrada la bóveda difusa.

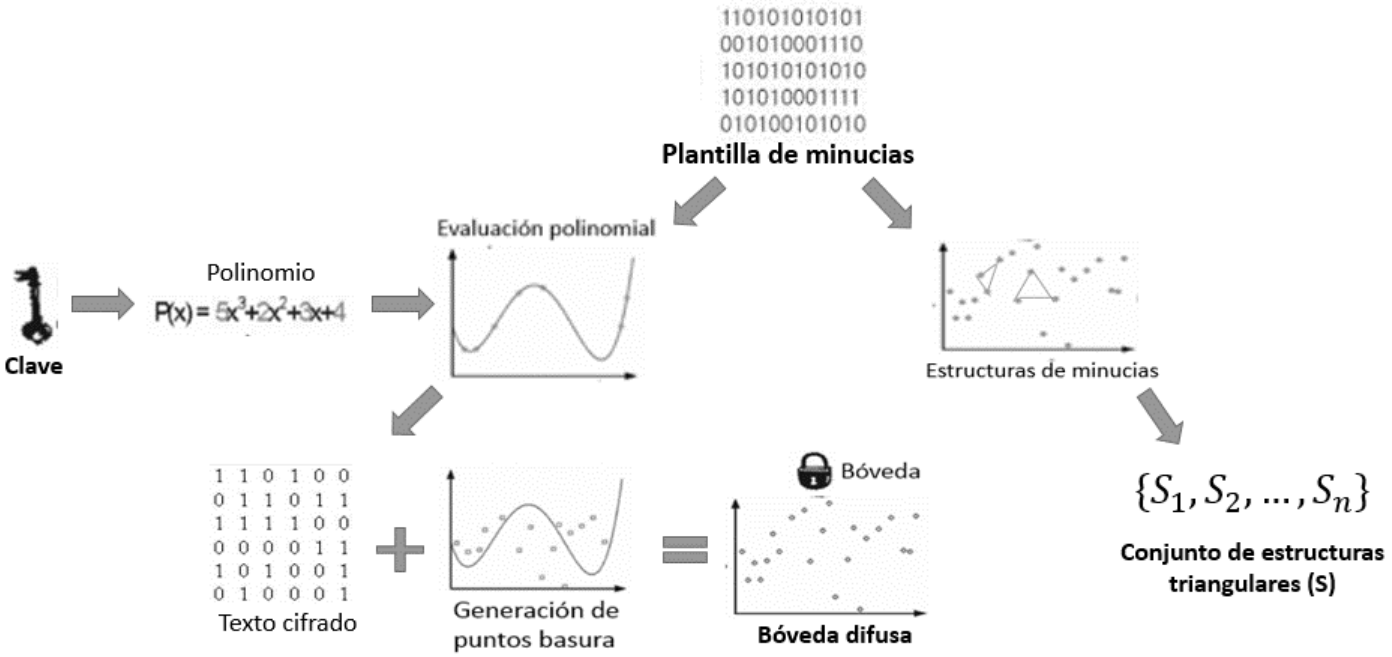

Figura 11. "Proceso de cifrado de la plantilla biométrica"

Comparación

Para realizar el proceso de comparación, una vez obtenida la plantilla de minucias de consulta, el sistema genera las estructuras triangulares S'_i correspondientes y realiza la búsqueda de al menos una coincidencia (34) entre estas y el conjunto de estructuras triangulares seleccionado de cada bóveda difusa. En caso de existir se alinean, aplicando rotación y traslación, al conjunto de puntos que representan las minucias de consulta para hacerlos corresponder, teniendo en cuenta el par punto-vector (p, v) de la estructura triangular coincidente del conjunto protegido correspondiente a una bóveda difusa. De lo contrario, en caso de no encontrarse coincidencia, se asume que la plantilla de minucias de consulta no corresponde al rasgo biométrico protegido. Con el conjunto alineado se procede a realizar la interpolación de Lagrange (58) sobre la bóveda difusa, obteniendo un polinomio q. A través del código de corrección de errores, con la clave dada por el polinomio q y la clave almacenada, si la coincidencia entre ellas es de al menos 2/3 del total de elementos de la clave almacenada entonces se asume que los datos de consulta pertenecen al mismo rasgo biométrico que el cifrado por la bóveda difusa.

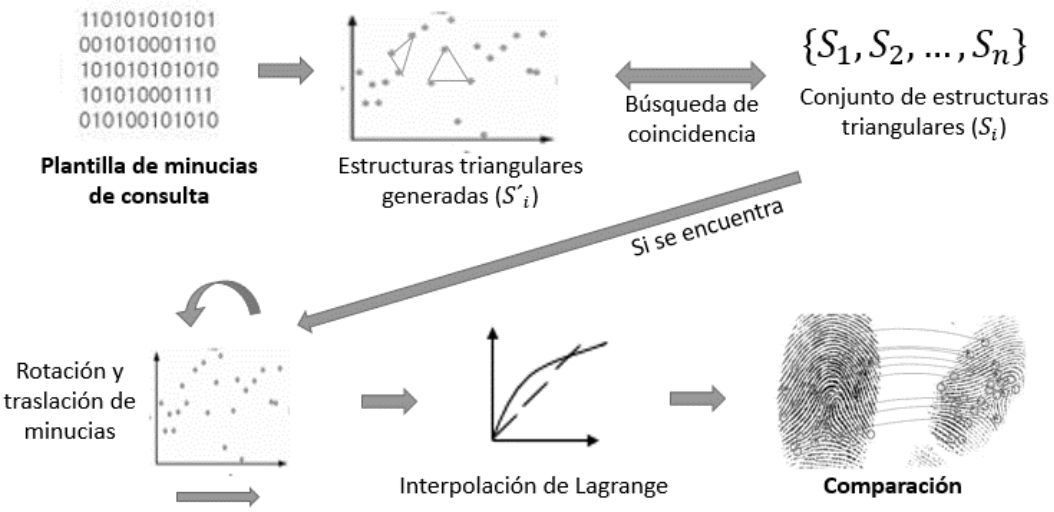

Figura 12. "Proceso de comparación de plantillas de minucias en el dominio cifrado"

2.3 **Modelo de dominio**

En el modelo de dominio que a continuación se muestra se evidencian los principales conceptos con los que se trabaja, ayudando a comprender el entorno y a definir las clases más significativas.

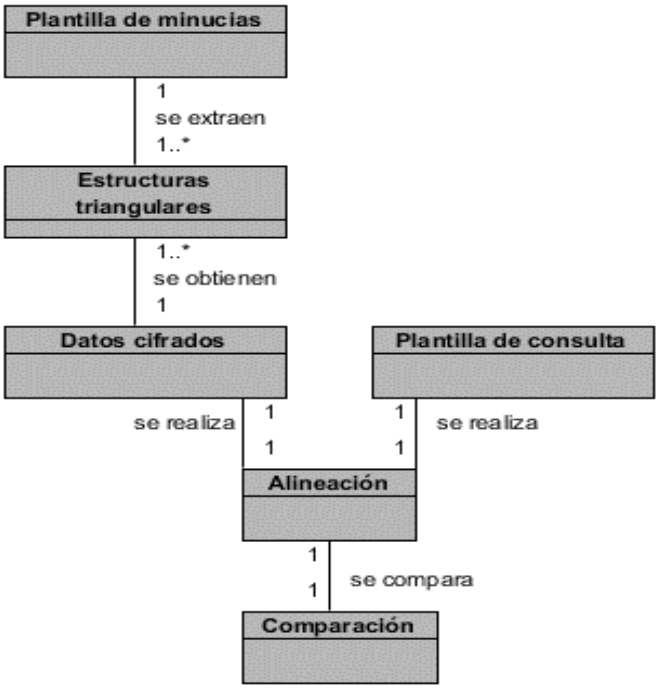

Figura 13. "Diagrama del modelo de dominio"

2.4 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.

2.4.1 Captura de requisitos

Es importante realizar la captura de los requisitos para la correcta realización de las siguientes 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.

2.4.1.1 Requisitos funcionales (RF)

RF1: Cargar la plantilla de minucias.

RF2: Generar tripletas de minucias.

RF3: Determinar las características identificativas de las tripletas de minucias.

RF4: Cifrar los datos.

RF5: Almacenar la bóveda difusa, un conjunto de datos pertenecientes a las estructuras triangulares y la clave.

RF6: Alinear la plantilla de minucias.

RF7: Comparar las plantillas en el dominio cifrado.

2.4.1.2 Requisitos no funcionales (RNF)

1. *Software*

- Multiplataforma.
- *Framework* de desarrollo: Qt 5.4.0

2. *Hardware*

- PC Dual Core o superior.
- CPU 1.3 GHZ o superior.
- 2 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.

2.4.2 Historias de Usuario (HU)

Las historias de usuario (HU) representan las funcionalidades que se desean que estén presentes en el sistema (59). La descripción de la historia de usuario cifrar los datos se muestra a continuación. El resto de las historias de usuario de encuentran descritas en los anexos (ANEXO#1).

Historia de Usuario	
Número: HU_4	**Usuario:** Sistema
Nombre historia: Cifrar los datos.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Dailín Martínez Pardo.	
Descripción: El sistema evaluará cada minucia en el polinomio P establecido a partir de una clave k generada aleatoriamente, obteniéndose para cada una de ellas sus evaluaciones en el polinomio atendiendo a los valores de x, y. Seguidamente se genera un conjunto de puntos basura, los cuales se mezclan aleatoriamente con los originales con el objetivo de protegerlos, quedando constituida así la bóveda difusa.	
Observaciones: -	

Tabla 1. Historia de usuario correspondiente a la funcionalidad Cifrar los datos.

2.4.3 Plan de iteraciones

Una vez descritas las HU se está en condiciones de realizar la planificación de la fase de implementación. Esta fase se dividirá en 2 iteraciones:

Iteración 1: En esta iteración se realizarán las historias de usuario que inciden en la generación de la bóveda difusa.

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.

2.4.3.1 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
	Generar tripletas de minucias	
	Determinar las características identificativas de las tripletas de minucias	
	Cifrar los datos	
	Almacenar la bóveda difusa, un conjunto de datos pertenecientes a las estructuras triangulares y la clave	
2	Alinear la plantilla de minucias	8
	Comparar las plantillas en el dominio cifrado	

Tabla 2. Plan de duración de las iteraciones.

2.4.4 Plan de entregas

Una vez concluida la elaboración de las diferentes HU se define el plan de entregas.

No. de entrega	Iteración	Fecha de entrega
1ra	1	24 de febrero del 2015
2da	2	28 de abril del 2015

Tabla 3. Plan de entrega.

2.5 Fase de diseño

La fase de diseño permite comprender el funcionamiento del sistema a implementar. Siguiendo las guías de la metodología XP, su función es crear una arquitectura de software que constituya el punto de partida para las actividades de implementación, donde se describan las clases existentes y la interacción entre ellas.
La metodología XP no requiere la representación del sistema mediante diagramas de clases utilizando el Lenguaje Unificado de Modelado (UML); en su lugar usa una técnica para representar clases, emplea tarjetas Clase-Responsabilidad-Colaboración (CRC) (40).

2.5.1 Tarjetas CRC

Las tarjetas CRC son una técnica utilizada en XP para diseñar la solución informática según el paradigma orientado a objetos. Las siglas CRC se refieren a *Class, Responsibilities* y *Collaborators* que son precisamente los elementos fundamentales de estas tarjetas. Para su confección se identifican las responsabilidades del *software*, los objetos que las ejecutan, así como aquellos objetos que colaboran en una determinada responsabilidad (60). A continuación se muestra en la tabla 4 la tarjeta CRC de la clase alineacion[13] perteneciente al componente de igual nombre. El resto de las tarjetas CRC se encuentran en los anexos (ANEXO#2).

Nombre de la clase: Alineacion	
Responsabilidades	**Colaboradores**
Alinear	1. Triangulo
	2. Estructuracion
Traslacion	1. Minucia
Rotacion	1. Minucia

Tabla 4. Tarjeta CRC correspondiente a la clase Alineacion.

[13] Los nombres de los componentes, clases y métodos que lleven tilde carecerán de esta, pues el IDE Qt Creator no permite el uso de signos de acentuación.

2.5.2 **Arquitectura del sistema**

La arquitectura de *software* es una vista del sistema que incluye los principales componentes del mismo, la conducta de estos según se percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión de este. Establece los fundamentos para que analistas, diseñadores, programadores y otros roles, trabajen en una línea común que permita alcanzar los objetivos del sistema, cubriendo todas las necesidades (61).

Para el desarrollo del producto se utilizará una **arquitectura basada en componentes** pues 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 (62). 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 (62).

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 genera la bóveda difusa y el de comparación que brinda la respuesta final según la correspondencia biométrica. Un componente denominado utilitarios, que agrupa las clases que son utilizadas por el resto de los componentes y el de alineación, el cual interviene tanto en el cifrado de la plantilla como en la comparación de esta con la plantilla de consulta (Ver figura 14).

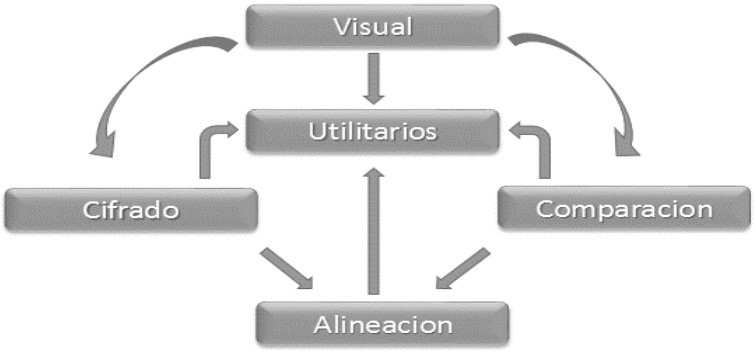

Figura 14. "Arquitectura basada en componentes"

En (38) los estilos 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 estilo 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 (62).

Una tubería (*pipeline*) es una arquitectura que conecta componentes computacionales (filtros) a través de conectores (*pipes*), de modo que las computaciones se ejecutan a la manera de un flujo. 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. Debido a su simplicidad y su facilidad para captar una funcionalidad, es una arquitectura mascota cada vez que se trata de demostrar ideas sobre la formalización del espacio de diseño arquitectónico, igual que el tipo de datos pila lo fue en las especificaciones algebraicas o en los tipos de datos abstractos (60).

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. El destino y los filtros se conectan con tubos que implementan el flujo de datos entre los pasos de procesamiento.

Cifrado

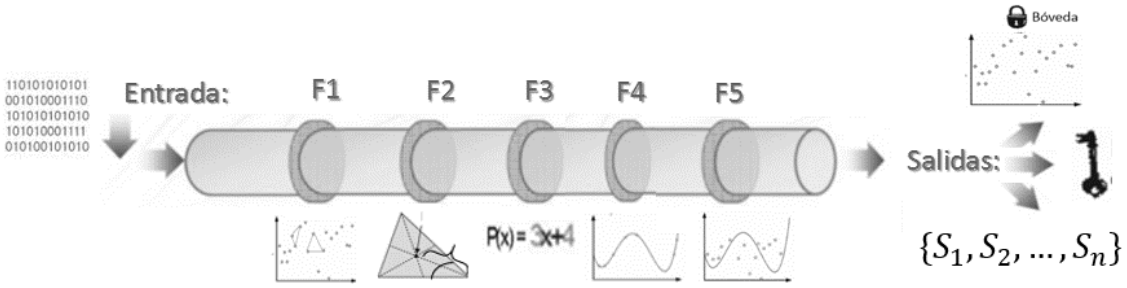

Figura 15. "Patrón tuberías y filtros en el proceso de cifrado de los datos"

1. Entrada: Plantilla de minucias cargada.
2. Filtro 1: Formación de las tripletas de minucias.
3. Filtro 2: Determinación de las características de las tripletas.
4. Filtro 3: Generación del polinomio a utilizar para la evaluación de las minucias.

5. Filtro 4: Evaluación polinomial.
6. Filtro 5: Generación de puntos basura.
7. Salidas: Unión del texto cifrado generado por la evaluación polinomial y los puntos basura (bóveda difusa), la clave y un conjunto de estructuras triangulares seleccionadas de la plantilla de minucias.

Comparación

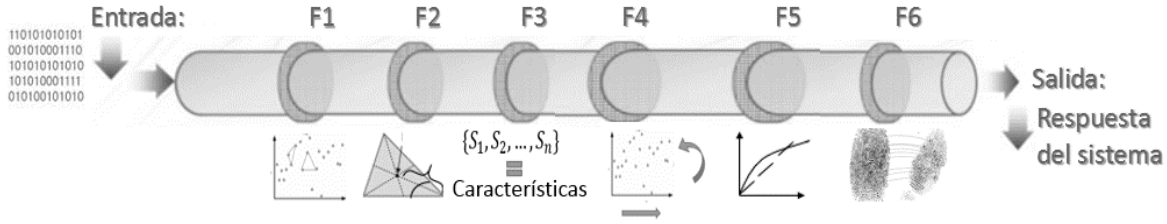

Figura 16. "Patrón tuberías y filtros en el proceso de comparación biométrica"

1. Entrada: Plantilla de minucias de consulta cargada.
2. Filtro 1: Formación de las tripletas de minucias.
3. Filtro 2: Determinación de las características identificativas de las tripletas.
4. Filtro 3: Búsqueda de coincidencias entre las características de las tripletas y el conjunto de estructuras triangulares seleccionado de la bóveda difusa.
5. Filtro 4: Rotación y traslación de las minucias de consulta.
6. Filtro 5: Interpolación de Lagrange.
7. Filtro 6: Comparación.
8. Salida: Respuesta que brinda el sistema sobre la autenticidad de la huella dactilar.

2.5.3 Patrones de diseño utilizados

En (63) un patrón de diseño provee un esquema para refinar los subsistemas o componentes de un sistema de *software* o las relaciones entre ellos. Describe la estructura comúnmente recurrente de los componentes en comunicación y resuelve un problema general de diseño en un contexto particular.

Se utilizaron los **patrones GRASP** (*General Responsibility Asignment Software Patterns*). Estos describen los principios fundamentales de la asignación de responsabilidades a objetos (61). Dentro de los patrones de diseño GRASP se emplearon:

- Experto en información: este patrón es utilizado en la clase Comparacion, entre otras, pues dado una situación, va a tener la información necesaria para proveer una solución. Los objetos presentes en ella se valen de su propia información para realizar la tarea asignada, conservando el encapsulamiento.
- Alta Cohesión: es utilizado en la clase Minucia, ente otras, pues permite asignar la mayor cantidad de responsabilidades a la clase y que esta siga siendo sencilla. Una clase con alta cohesión es útil porque es fácil darle mantenimiento, entenderla y reutilizarla.
- Creador: se emplea en la clase Cifrado, entre otras, pues este patrón es el encargado de identificar quién es el responsable de la creación de los objetos.
- Bajo acoplamiento: en la clase Triangulo implementada se evidencia el uso de este patrón, el cual se enfoca en el empleo de clases lo menos ligadas entre sí, de forma tal que en caso de ocurrir una modificación en alguna, esto tenga la mínima repercusión en el resto de las clases.
- Controlador: se emplea en las clases Cifrado y Comparacion, pues este patrón plantea el uso de determinadas clases que sean las encargadas de dirigir todo el proceso.

Dentro de los **patrones GoF** (*The Gang of Four*), los cuales ayudan a la construcción de un *software* basado en la reutilización, se empleó el patrón ***Prototype*** (Prototipo). 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.

Ejemplo: nombreArchivoP=QFileDialog::getOpenFileName(this, tr("Seleccione la plantilla de minucias"), nombreArchivoP, tr("Plantilla de minucias Files(*.temp)"));

2.6 Conclusiones parciales

La propuesta de solución definida sirvió de guía para la implementación del componente. Las historias de usuario y las tarjetas CRC permitieron un mejor dominio del tema para lograr una correcta implementación de cada uno de los requisitos. La elaboración del plan de iteraciones y de entrega posibilitó organizar el trabajo y definir la duración de las tareas. La arquitectura basada en componentes y el patrón arquitectónico tuberías y filtros permitieron el desarrollo del componente con un correcto uso de los patrones de diseño para lograr un producto de calidad.

3 CAPÍTULO III: Implementación y pruebas

3.1 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.

3.2 Estándar de codificación

Para lograr una mejor comprensión del código es necesario definir reglas que proporcionen un criterio específico para la creación de los nombres de los métodos, clases y variables. A continuación se describe el estándar empleado en la implementación del componente.

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	Reglas de nombres	Ejemplos
Variables	Todas las variables comenzarán con minúsculas y la primera letra a partir de la segunda palabra, las cuales irán unidas la primera en caso de tener, comenzará con mayúscula.	indiceTrianguloCosulta

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.	CoincidenciaEstructural
Clases o Interfaces	Los nombres de las clases o interfaces deben tener la primera letra de cada palabra con mayúscula.	Minucia, Cifrado
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 "_".	PI

Tabla 5. Convenciones de nombres.

3.3 Tareas de ingeniería

En la etapa de planificación se definieron las historias de usuario a implementar en cada iteración, pues XP define que la implementación de un sistema debe realizarse iterativamente. Para lograr esto se deben descomponer las HU en tareas de la ingeniería y ser asignadas a los programadores encargados de implementarlas en la iteración correspondiente. En la siguiente tabla se describen las tareas por HU correspondientes a la primera iteración. Las tareas de la ingeniería de la segunda iteración se encuentran en los anexos (ANEXO#3).

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.
Generar tripletas de minucias	1. Construcción de todos los triángulos posibles, asegurando que el orden de los vértices sea en sentido anti-horario. 2. Adición de los triángulos a la lista de tripletas si cumplen la condición de tener un lado más corto que los dos restantes.
Determinar las características identificativas de las tripletas de minucias	1. Cálculo del baricentro. 2. Cálculo de la longitud de los lados. 3. Cálculo de los ángulos que forman cada uno de los lados respecto al eje de las abscisas. 4. Cálculo de los ángulos relativos de cada vértice (minucia) respecto al lado correspondiente. 5. Determinación del vector desplazamiento que se forma entre el baricentro y el vértice opuesto al lado más corto.
Cifrar los datos	1. Generación de una clave de longitud k. 2. Definición del polinomio a partir de la clave k. 3. Evaluación de cada minucia en el polinomio. 4. Generación de los puntos basura. 5. Mezcla de los puntos basura con los puntos originales.
Almacenar la bóveda difusa, un conjunto de datos pertenecientes a las estructuras triangulares y la clave.	1. Pedido de la dirección donde se almacenará la bóveda difusa, la clave y los datos de las estructuras triangulares seleccionadas.

	2. Almacenamiento de la bóveda difusa y los datos en un fichero con extensión .bv y la clave en otro con extensión .key.

Tabla 6. Tareas de la ingeniería en la primera iteración.

3.4 Diagrama de componentes

El diagrama de componentes permitirá un mejor entendimiento del flujo de trabajo de la implementación. Este 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.

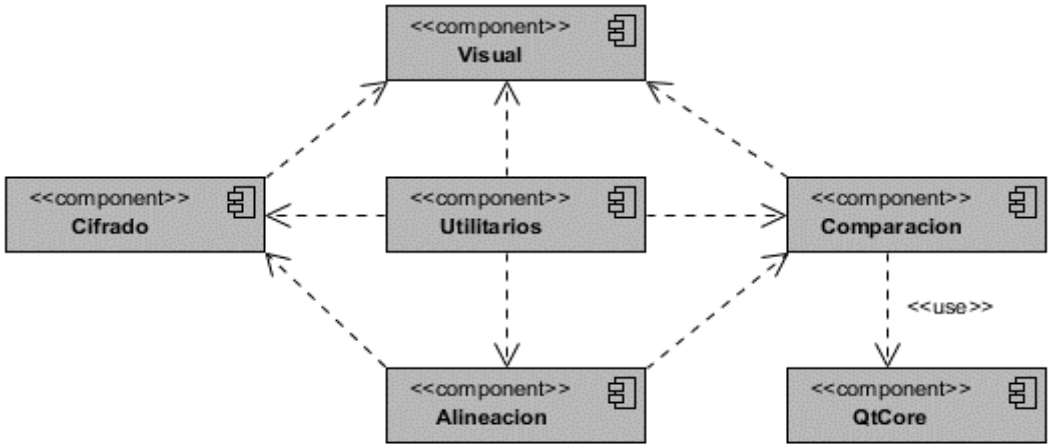

Figura 17. "Diagrama de componentes"

Descripción de los componentes:

Componente	Propósito
Visual	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.
Cifrado	Genera la bóveda difusa.

Comparacion	Brinda la respuesta final según la correspondencia biométrica.
Utilitarios	Agrupa las clases que son utilizadas por el resto de los componentes.
QtCore	Forma parte de las librerías del *framework* que se utiliza en el proyecto.
Alineacion	En el caso del cifrado genera las estructuras de minucias y en la comparación alinea las estructuras almacenadas con una que recibe por parámetros.

Tabla 7. Descripción de los componentes del sistema.

3.5 Pruebas

La realización de pruebas de *software* es fundamental para detectar errores y verificar la calidad del producto teniendo como base el control del cumplimiento de los requisitos. Para validar el correcto funcionamiento del componente se realizarán pruebas de validación, unidad y efectividad.

3.5.1 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 (38).

Pruebas de caja negra:

Las pruebas de caja negra son las que se realizan sobre la interfaz del sistema. A través de casos de prueba se demuestra que las funcionalidades del componente operan de forma correcta. A continuación se muestran los casos de pruebas correspondientes a las historias de usuario cargar la plantilla de minucias y almacenar la bóveda difusa, un conjunto de datos pertenecientes a las estructuras triangulares y la clave. El resto de los casos de pruebas se encuentran en los anexos (ANEXO#4).

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: Dailín Martínez Pardo	
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 definida en los anexos (ANEXO#5).	
Entrada/Pasos de ejecución: Nombre del fichero, que incluye la dirección.	
Resultado esperado: La plantilla de minucias cargada.	
Evaluación: Satisfactoria	

Tabla 8. Caso de prueba de la funcionalidad: Cargar la plantilla de minucias.

Casos de prueba	
Código Caso de Prueba: 2	**Nombre Historia de Usuario:** Almacenar la bóveda difusa, un conjunto de datos pertenecientes a las estructuras triangulares y la clave.
Nombre de la persona que realiza la prueba: Dashiel Hondarez Laza	
Descripción de la prueba: Una vez generada la bóveda difusa se pedirá la dirección donde se almacenará esta junto a la clave y los datos de las estructuras triangulares seleccionadas.	
Condiciones de ejecución: La bóveda difusa y los datos se almacenarán en un fichero con extensión .bv y la clave en otro con extensión .key.	
Entrada/Pasos de ejecución: Dirección de almacenamiento.	
Resultado esperado: La bóveda difusa, un conjunto de datos pertenecientes a las estructuras triangulares y la clave almacenados.	

Evaluación: Satisfactoria

Tabla 9. Caso de prueba de la funcionalidad: Almacenar la bóveda difusa, un conjunto de datos pertenecientes a las estructuras triangulares y la clave.

Se realizaron 3 iteraciones de estas pruebas, encontrándose en la primera iteración 7 no conformidades, en la segunda 2 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.

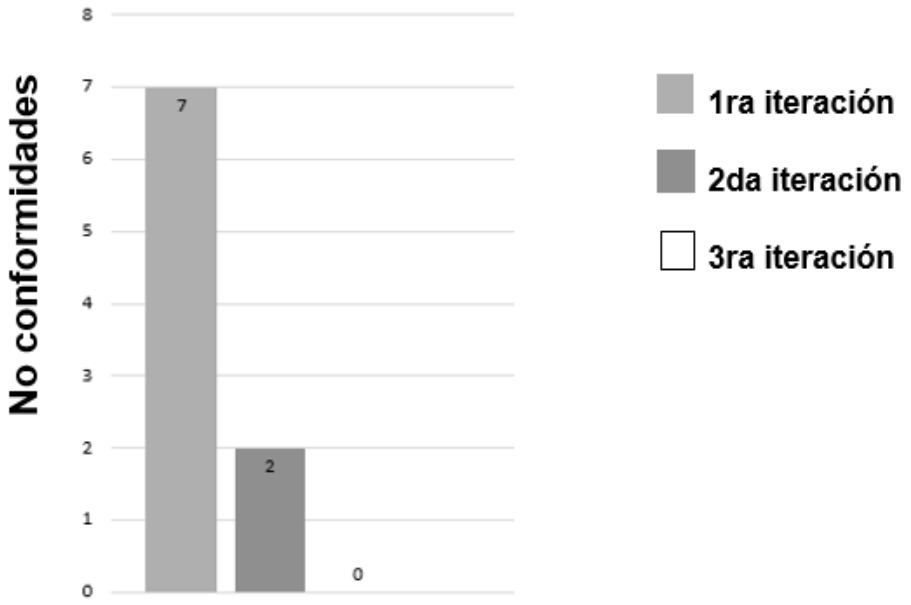

Figura 18. "Resultados de las pruebas de caja negra"

3.5.2 **Pruebas de unidad**

Las pruebas de unidad son las que van enfocadas a los elementos más pequeños del *software*. Son aplicables a funcionalidades para verificar que los flujos de control y de datos están cubiertos y funcionan como se espera (38). Las pruebas de unidad están orientadas a caja blanca.

Pruebas de caja blanca:

Las pruebas de caja blanca requieren del conocimiento de la estructura interna del programa y son derivadas a partir de las especificaciones internas del diseño o el código. Se examinan los caminos lógicos del sistema exponiendo que los casos de prueba se realizan juntos, definidos por condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado

o mencionado (64). La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes.

Prueba del camino básico: Permite al diseñador de casos de pruebas obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución (64).

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. Puede calcularse de tres formas alternativas (64):

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

Realización de pruebas de caja blanca a las funcionalidades GenerarK y EvaluarP.

- **Funcionalidad:** GenerarK

```
void Cifrado::GenerarK(){
 int hora=time(NULL); //1
 // semilla de rand();
 srand(hora); //1
 int numero; //1
 int total=ElementosK/2; //1
 for (int var = 0; var < total; ++var) //2
 {
 numero =1+rand()%9; //3
 claveX+=QString::number(numero); //3
 numero =1+rand()%9; //3
 claveY+=QString::number(numero); //3
 }
 if(ElementosK%2!=0) //4
 {
 numero =rand()%10; //5
 claveY+=QString::number(numero); //5
 }
 claveTotal=claveX+claveY; //6
```

```
}
```

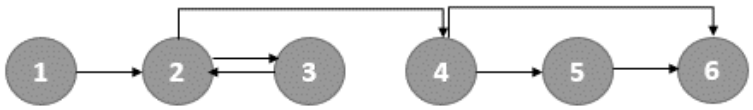

Figura 19. "Grafo de flujo: Funcionalidad GenerarK"

Complejidad ciclomática:

Fórmula 1:

V (G) = (A (Aristas) – N (Nodos)) + 2

V (G) = 7 – 6 + 2 = 3

Fórmula 2:

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

V (G) = 2 + 1 = 3

Fórmula 3:

V (G) = R = 3

Caminos independientes obtenidos:

1-2-4-5-6

1-2-3-2-4-6

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

- **Funcionalidad:** EvaluarP

```
double Polinomio::EvaluarP(int X){
 int CoeficienteX; //1
 double retorno=0; //1
 int Exponente=3; //1
 for (int i = 0; i < 4; i+=1) //2
      {
       CoeficienteX=Coeficientes[i]; //3
 <coeficiente>*x^<Exponente> //3
       Monomio *Mon=new Monomio(CoeficienteX, Exponente, X); //3
 if(i==0) //4
       retorno=retorno+Mon->Evaluar(); //5
       else //6
       retorno=retorno-Mon->Evaluar();//6
       delete Mon; //7
       if(Exponente>0) //8
```

```
 Exponente--; //9
 }
 return retorno; //10
}
```

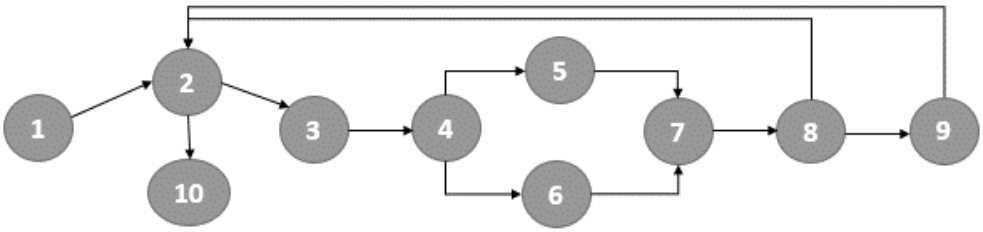

Figura 20. "Grafo de flujo: Funcionalidad EvaluarP"

Complejidad ciclomática:

Fórmula 1:

V (G) = (A (Aristas) – N (Nodos)) + 2

V (G) = 12 – 10 + 2 = 4

Fórmula 2:

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

V (G) = 3 + 1= 4

Fórmula 3:

V (G) = R = 4

Caminos independientes obtenidos:

1-2-3-4-5-7-8-2-10

1-2-3-4-5-7-8-9-2-10

1-2-3-4-6-7-8-2-10

1-2-3-4-6-7-8-9-2-10

Se realizaron 3 iteraciones de estas pruebas, encontrándose en la primera iteración 8 no conformidades, en la segunda 3 y en la tercera ninguna. Entre las no conformidades identificadas estuvo la incorrecta generación de los puntos basura y en la construcción de las estructuras triangulares no se validó que el sentido de los vértices fuera anti-horario.

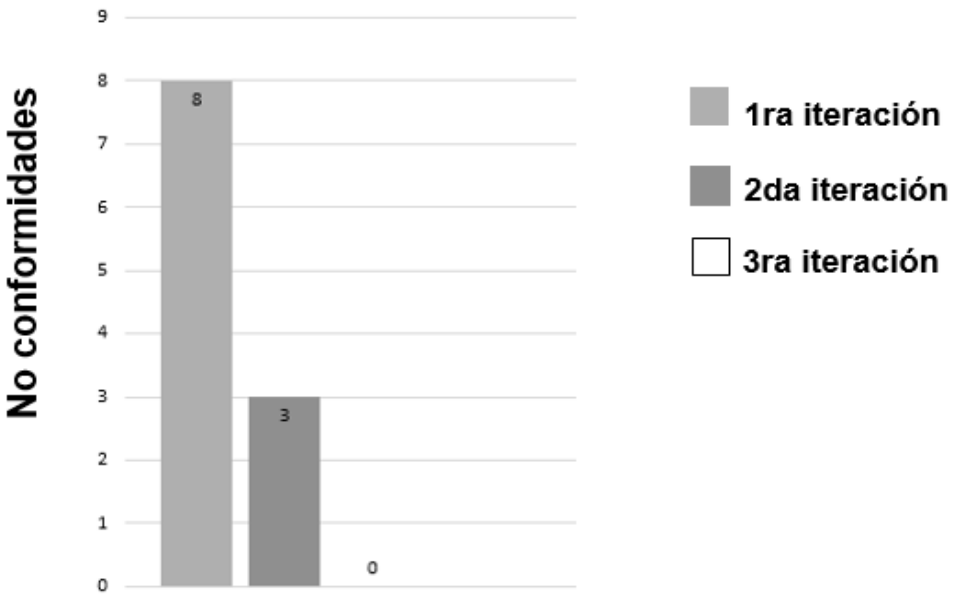

Figura 21. "Resultados de las pruebas de caja blanca"

3.5.3 **Pruebas de efectividad**

Evaluación del rendimiento del componente:

Para evaluar el rendimiento es necesario analizar la precisión en el proceso de identificación. Para la realización de estas pruebas se utilizaron las 4 bases de datos del FVC2000[14]. Cada base de datos está compuesta por 80 imágenes de huellas dactilares distribuidas en 8 tomas diferentes por cada dedo de la mano (10 dedos). 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).

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 (tomándose solo 8 dedos).

[14] **FVC2000:** es la Primera Competencia Internacional de Algoritmos de Verificación de Huellas Dactilares.

De las 6 muestras seleccionadas por cada dedo una se utilizará para la construcción de la bóveda difusa y 5 para la comparación. Los parámetros para determinar dicha precisión de manera cuantitativa son las tasas de falsos positivos (FMR) y falsos negativos (FNMR) (1).

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.

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.

Teniendo en cuenta la tipificación de las bases de datos se realizaron 2016 comparaciones para la FMR y 240 comparaciones para la FNMR. En la tabla 10 se muestra el resultado de dichas pruebas.

Base de datos del FVC 2000	FMR	FNMR
BD_1	0	0.4
BD_2	0	0.6
BD_3	0	0.5
BD_4	0	0.5

Tabla 10. Resultados de las pruebas de efectividad.

En el ANEXO#6 (figuras 23 y 24) se muestra el resultado de estas pruebas para una base de datos y tomándose solo una bóveda de cada dedo. Se realizaron 40 comparaciones para la FNMR y 280 comparaciones para la FMR.

3.6 Conclusiones parciales

El diseño del diagrama de componentes permitió un mejor entendimiento del flujo de trabajo en la etapa de implementación. Las pruebas de *software* realizadas a la aplicación permitieron identificar las no conformidades y corregirlas, asegurando en mayor grado una solución de calidad. Las pruebas realizadas al componente implementado arrojaron buenos resultados, permitiendo identificar a una persona en el dominio cifrado.

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 bóveda difusa.
- 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 estilo 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 se recomienda:

- Desarrollar otros algoritmos de selección de características identificativas para realizar el cifrado y la comparación de los datos.
- Desarrollar una estrategia de comparación que maximice el uso de los datos almacenados para disminuir las tasas de error obtenidas.

REFERENCIAS BIBLIOGRÁFICAS

1. Maltoni, D., y otros. *Handbook of Fingerprint Recognition.* New York : Springer-Verlag, 2009. 978-1-84882-253-5.

2. Sanz, Gustavo Francisco. *Desarrollo de un sistema de reconocimiento de huella dactilar para aplicaciones Match-On-Card.* s.l. : Universidad Autónoma de Madrid, Escuela Politécnica Superior, 2009.

3. Dahiya, Neha y Kant, Chander. *Biometrics Security Concerns.* Haryana, India : Second International Conference on Advanced Computing, 2012. 978-0-7695-4640-7.

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

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

6. Domínguez, Sara Carballo. *Ataques indirectos a sistemas de reconocimiento de huella dactilar basados en los tiempos de comparación algorítmica.* Madrid : Escuela politécnica superior, 2009.

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

8. Cappelli, Raffaele, y otros. *Fingerprint Image Reconstruction from Standard Templates.* s.l. : IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, septiembre 2007. págs. 1489-1504. Vol. 29.

9. Gordon, Jeffrey I. Forensic identification using skin bacterial communities. [En línea] febrero de 2010. [Citado el: 27 de enero de 2015.] http://www.pnas.org/content/early/2010/03/01/1000162107.

10. Cruz, Aradí rosales. Clasificación de huellas digitales mediante minucias. [En línea] 2009. [Citado el: 2015 de enero de 26.] http://ccc.inaoep.mx/~esucar/Clases-mgp/Proyectos/reporte_modelos_huellas.pdf.

11. Sutcu, Yagiz, T. Sencar, Husrev y Memon, Nasir. *Geometric Transformation to Protect Minutiae-Based Fingerprint Templates.* New York : s.n., 2007.

12. Ribalta, Albert Solé. *Seguridad en los sistemas biométricos.* Catalunya, España : FUOC. 00200719.

13. Narayanan, Radha y Subramanian, Kathikeyan. *An Efficient Secure Biometric System with Non-Invertible Gabor Transform.* s.l. : IJCSI International Journal of Computer Science Issues, 2011. 1694-0814.

14. Nagar, Abhishek y K. Jain, Anil. *ON THE SECURITY OF NON-INVERTIBLE FINGERPRINT TEMPLATE TRANSFORMS.* Michigan : Department of Computer Science and Engineering, Michigan State University, 2010.

15. N. Radha, S. Karthikeyan. *AN EVALUATION OF FINGERPRINT SECURITY USING NONINVERTIBLE BIOHASH.* s.l. : International Journal of Network Security, 2011.

16. F. Farooq, R. Bolle, T. Jea, N. Ratha. *Anonymous and revocable fingerprint recognition.* s.l. : IEEE CVPR, 2007.

17. Kumar, Nitesh. *CANCELABLE FINGERPRINT TEMPLATES.* Indian Academy of Science : Department of Mathematics, IIT Kharagpur, 2012.

18. *Cancelable fingerprint templates using minutiae-based bit-strings.* Lee, Chulhan y Kim, Jaihie. 33, s.l. : Journal of Network and Computer Application, 2010, págs. 236-246.

19. *Biohashing: two factor authentication featuring fingerprint data and tokenised random number.* Teoh Beng Jin, Andrew, Ngo Chek Ling, David y Goh, Alwyn. Malaysua : The Journal of the pattern recognition society., 2004. 2245-2255.

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

21. Chikkerur. *Online fingerprint verification system.* Univ of New York : s.n., 2005.

22. *Fingerprint reference-point detection.* M. Liu, X. Jiang, A. Kot. s.l. : AURASIP Journal, 2005.

23. Juels, Ari y Sudán, Madhu. *A fuzzy Vault Sheme.* 2002.

24. Anil K., Jain, Ross, Arun y Prabhakar, Salil. *An introduction to biometric recognition.* s.l. : IEEE Trans. Circuits Syst. Video Tech., 2006.

25. URBE. Virtual URBE. [En línea] [Citado el: 11 de febrero de 2015.] http://virtual.urbe.edu/tesispub/0092449/cap02.pdf.

26. *An alignment-free fingerprint cryptosystem based on fuzzy vault scheme.* Li, Peng, y otros. 3, 2009, Vol. 22, págs. 207-220.

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

28. ***Securing Fingerprint Template: Fuzzy Vault With Helper Data.*** **Uludag, U., Jain, A.K. New York, USA : Proc. CVPR Workshop on Privacy Research In Vision, 2006.**

29. Sorroza, Gilberto Emilio Hernández. ***Obtención de características deltas, cores y encierros en una huella dactilar.*** **Huajuapan de León, OAX : s.n., 2002.**

30. Boonchaiseree, N., Areekul, V. ***Focal Point Detection Based on Half Concentric Lens Model for Singular Point Extraction in Fingerprint.*** **s.l. : Proceedings of International Conference on Biometrics, 2009.**

31. ***Automatic Secure Fingerprint Verification System Based on Fuzzy Vault Scheme.*** **Yang, S., Verbauwhede, I. Philadelphia, USA : Proc. IEEE International Conference on Acoustics, Speech, and Signal, 2005, Vol. 5.**

32. ***Fingerprint Classification Using Orientation Field Flow Curves.*** **Dass, S.C., Jain, A.K. Kolkata, India : Proc. Indian Conference on Computer Vision, Graphics and Image Processing, 2004, págs. 650-655.**

33. ***A Fingerprint Cryptosystem based on Minutiae Phase Spectrum.*** **Nandakumar, K. Seattle, USA : Proc. of Second IEEE Workshop on Info. Forensics & Security, 2010.**

34. Jeffers, Jason y Arakala, Arathi. ***Fingerprint alignment for a minutia-based fuzzy vault.*** **Melbourne, Australia : s.n., 2007.**

35. Tinoco Gómez, Oscar, Rosales López, Pedro Pablo y Salas Bacalla, Julio. ***Criterios de selección de metodologías de desarrollo de software.*** **Revista de la Facultad de Ingeniería Industrial : s.n., 2010. 1560-9146.**

36. Figueroa, Roberth G., Solís, Camilo J. y Cabrera, Armando A. ***Metodologías tradicionales vs metodologías ágiles.*** **Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación. : s.n.**

37. Romero, Hermenegildo. SlideShare. [En línea] 2009. [Citado el: 06 de febrero de 2015.] http://es.slideshare.net/MeneRomero/metodologias-de-desarrollo.

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

39. Valdés, José Luis Cendejas. eumed.net. ***Enciclopedia Virtual.*** **[En línea] [Citado el: 06 de febrero de 2015.] http://www.eumed.net/tesis-doctorales/2014/jlcv/software.htm.**

40. Letelier , Patricio y Penadés, Mª Carmen. ***Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP).*** **España : Universidad Politécnica de Valencia. : s.n.**

41. IBM. Unified Modeling Language (UML). [En línea] [Citado el: 7 de febrero de 2015.] http://www-01.ibm.com/software/rational/uml/.

42. Alegsa, Leandro. Alegsa. *Diccionario de Informática y Tecnología.* [En línea] 2015. [Citado el: 7 de febrero de 2015.] http://www.alegsa.com.ar/Dic/uml.php.

43. Rumbaugh, James y Jacobson, Ivar. *El lenguaje Unificado de Modelado, Manual de Referencia.* 2000. 8478290370.

44. Larmn, Craig. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 2004.

45. Jacobson, Booch Rumbaugh. *Proceso unificado en el desarrollo de software.*

46. Nobrega, Maria De. Herramientas CASE: Rational Rose. [En línea] http://curso_sin2.blogia.com/2005/060401-herramientas-case-rational-rose.-por-maria-de-nobrega.php.

47. Visual Paradigm. [En línea] [Citado el: 17 de enero de 2015.] http://www.visual-paradigm.com./.

48. Baeza, Pablo Nicolás. Visual Paradigm DB Visual ARCHITECT SQL. [En línea] [Citado el: 7 de febrero de 2015.] http://www.docstoc.com/docs/96492173/Visual-Paradigm-Studio.

49. Lenguajes de Programación. [En línea] 2009. [Citado el: 12 de diciembre de 2014.] http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml.

50. *Comparative Programming Languages.* Wilson, Leslie Blackett. s.l. : Addison-Wesley, 1993. 0-201-56885-3..

51. AmericaTI. Ventajas y Desventajas: Comparación de los Lenguajes C, C++ y Java. [En línea] http://www.americati.com/doc/ventajas_c.pdf.

52. Java. [En línea] [Citado el: 26 de enero de 2015.] http://arantxa.ii.uam.es/~castells/docencia/poo/2-java-esp.pdf.

53. *Iniciación a la programación en C#: un enfoque práctico.* Cerezo, Yolanda López. s.l. : Delta Publicaciones, 2006.

54. Microsoft. Introducción al lenguaje C# y .NET Framework. [En línea] [Citado el: 23 de febrero de 2015.] http://msdn.microsoft.com/es-es/library/z1zx9t92%28v=VS.80%29.aspx.

55. Salavert Ramos, Isidro y Pérez Lozano, María Dolores. *Entornos de Desarrollo Integrado. Ingeniería Del Software Y Bases de Datos: Tendencias Actuales.* Univ de Castilla La Mancha : s.n., 2000.

56. Entornos integrados de desarrollo. [En línea] [Citado el: 23 de enero de 2015.] http://carlosblanco.pro/2012/04/entornos-desarrollo-integrado-introduccion..

57. Nokia Corporation. Qt Reference Documentation. [En línea] 2008-2012. [Citado el: 25 de marzo de 2015.] http://doc.qt.digia.com/qtcreator-extending/.

58. Álvarez Blanco, Manuel, Guerra Hernández, Alfredo y Lau Fernández, Rogelio. *Matemática numérica.* Ciudad de la Habana : Félix Varela, 2007. Vol. 1. 978-959-07-0571-7.

59. Suaza, Katerine Villamizar. *Definición de equivalencias entre historias de usuario y especificaciones en UN-LENCEP para el desarrollo ágil de software.* Medellín, Colombia : Departamento de Ciencias de la Computación y de la Decisión, Universidad Nacional de Colombia, 2013.

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

61. Departamento Central de Ingeniería de Software. *Flujo de trabajo Captura de requisitos. Modelo de Negocio.* Ciudad de la Habana, Universidad de Ciencias Informáticas : s.n., 2004.

62. Camacho, Erika, Cardeso, Fabio y Nuñez, Gabriel. *Arquitecturas de Software.* abril 2004.

63. Sommerville, Ian. *Ingeniería de Software.* 2005.

64. Letelier, Patricio. *Pruebas del Software.* Universidad Politécnica de Valencia : Departamento Sistemas Informáticos y Computación.

GLOSARIO DE TÉRMINOS

- **Algoritmo:** Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema. Un conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos que no generen dudas a quien lo ejecute.
- ***Framework:*** Estructura de artefactos o módulos concretos con base en la que otro proyecto de *software* puede ser desarrollado.
- **Umbral:** Valor mínimo a partir del cual se considera que un elemento pertenece a una clase.
- **Epidermis:** Capa superficial de la piel**.**
- **Discretización:** Aproximación de la solución de un problema representándolo en términos de un conjunto discreto de elementos.
- **Multiplataforma:** Se refiere a los programas informáticos que pueden funcionar en diversas plataformas.
- **Tasa de falso rechazo** (*False Non-Match Rate*, *FNMR*): Se entiende como la probabilidad de que el sistema de autenticación rechace a un usuario legítimo porque no es capaz de identificarlo correctamente.
- **Tasa de falsa aceptación** (*False Match Rate*, *FMR*): Es la probabilidad de que el sistema autentique correctamente a un usuario ilegítimo.
- **Patrones GRASP** (*General Responsibility Asignment Software Patterns*): Patrones generales de *software* para asignar responsabilidades para el diseño del componente.
- **Entropía:** Es la aleatoriedad recogida por un sistema operativo o una aplicación para su uso en criptografía.

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 desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Dailín Martínez Pardo.	
Descripción: El sistema solicita la dirección del fichero a cargar, la cual es especificada por el usuario.	
Observaciones: -	

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

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: Dailín Martínez Pardo.	
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 12. 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: Dailín Martínez Pardo.	
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 13. Historia de usuario correspondiente a la funcionalidad Determinar las características identificativas de las tripletas de minucias.

Historia de Usuario	
Número: HU_5	**Usuario:** Sistema
Nombre historia: Almacenar la bóveda difusa, un conjunto de datos pertenecientes a las estructuras triangulares y la clave.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1

Programador responsable: Dashiel Hondarez Laza.
Descripción: El sistema solicitará la dirección donde se desea almacenar la bóveda difusa y los datos de las estructuras triangulares seleccionadas y los guardará en esta. Además almacenará un archivo que contendrá la clave con la que fue cifrada la bóveda difusa.
Observaciones: La bóveda difusa y los datos de las estructuras triangulares se almacenarán en un fichero con extensión .bv y la clave en otro con extensión .key.

Tabla 14. Historia de usuario correspondiente a la funcionalidad Almacenar la bóveda difusa, un conjunto de datos pertenecientes a las estructuras triangulares y la clave.

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: Dashiel Hondarez Laza.	
Descripción: El sistema intentará encontrar al menos una coincidencia entre las estructuras triangulares generadas de los datos de consulta y el conjunto de estructuras triangulares de cada bóveda difusa. En caso de existir se alinean, aplicando rotación y traslación, al conjunto de puntos que representan las minucias de consulta para hacerlos corresponder, teniendo en cuenta el par punto-vector (p, v) de la estructura triangular coincidente del conjunto protegido correspondiente a una bóveda difusa. 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 15. 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: Dashiel Hondarez Laza.	
Descripción: El sistema realizará la interpolación de Lagrange sobre la bóveda difusa, obteniendo un polinomio q. A través del código de corrección de errores, con la clave dada por el polinomio q y la clave almacenada, si la coincidencia entre ellas es de al menos 2/3 del total de elementos de la clave almacenada entonces se asume que los datos de consulta pertenecen al mismo rasgo biométrico que el cifrado por la bóveda difusa.	
Observaciones: -	

Tabla 16. Historia de usuario correspondiente a la funcionalidad Comparar las plantillas de minucias en el dominio cifrado.

ANEXO#2: Tarjetas CRC

Componente: Alineacion

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

Tabla 177. Tarjeta CRC correspondiente a la clase Estructuracion

Componente: Cifrado

Nombre de la clase: Cifrado	
Responsabilidades	**Colaboradores**
Evaluar	3. Minucia
	4. Polinomio
	5. Estructuracion
Escribir	2. Minucia
	3. Triangulo
GenerarK	-
GenerarBasura	1. Minucia
	2. Polinomio

Tabla 188. Tarjeta CRC correspondiente a la clase Cifrado.

Nombre de la clase: Monomio	
Responsabilidades	**Colaboradores**
Evaluar	-

Tabla 19. Tarjeta CRC correspondiente a la clase Monomio.

Nombre de la clase: Polinomio	
Responsabilidades	**Colaboradores**
LLenarArregloCoeficientes	-
EvaluarP	1. Monomio

Tabla 20. Tarjeta CRC correspondiente a la clase Polinomio.

Componente: Comparacion

Nombre de la clase: Comparacion

Responsabilidades	Colaboradores
ReordenacionCiclicaEspecial	1. Triangulo
CoincidenciaEstructural	1. Triangulo
BuscarCoincidenciaEstructural	1. Alineacion
VerificarRedundancia	-
CodigoCorreccionErrores	1. Minucia
Lagrange	-
Monomio2	-

Tabla 21. Tarjeta CRC correspondiente a la clase Comparacion.

Componente: Utilitarios

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

Tabla 22. Tarjeta CRC correspondiente a la clase Triangulo.

Componente: Visual

Nombre de la clase: Mainwindow	
Responsabilidades	**Colaboradores**
LeerPlantilla	1. Minucia
LeerBoveda	1. Minucia
	2. Triangulo
LeerClave	-

Tabla 23. Tarjeta CRC correspondiente a la clase Mainwindow.

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 que contiene la bóveda difusa y los datos pertenecientes a las estructuras triangulares. 3. Reordenación cíclica. 4. Búsqueda de coincidencias entre las estructuras triangulares. 5. 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.	6. Realización de la Interpolación de Lagrange. 7. Búsqueda de coincidencias entre la clave original y la extraída del polinomio devuelto por la interpolación realizada.

Tabla 24. Tareas de la ingeniería en la segunda iteración.

ANEXO#4: Casos de pruebas de validación

Casos de prueba	
Código Caso de Prueba: 3	**Nombre Historia de Usuario:** Generar tripletas de minucias.
Nombre de la persona que realiza la prueba: Dailín Martínez Pardo	
Descripción de la prueba: Se seleccionan todas las combinaciones posibles de tres minucias.	
Condiciones de ejecución: Que la tripleta formada cumpla con la condición de tener un lado menor.	
Entrada/Pasos de ejecución: Plantilla de minucias cargada.	
Resultado esperado: Formación de las estructuras triangulares.	
Evaluación: Satisfactoria	

Tabla 25. Caso de prueba de la funcionalidad: Generar tripletas de minucias.

Casos de prueba	
Código Caso de Prueba: 4	**Nombre Historia de Usuario:** Comparar las plantillas en el dominio cifrado.
Nombre de la persona que realiza la prueba: Dashiel Hondarez Laza	
Descripción de la prueba: Se alinean las plantillas, se realiza la interpolación y se compara si la coincidencia entre las claves es de al menos el 60%.	

Condiciones de ejecución: La plantilla de consulta, la bóveda difusa y la clave deben estar correctamente estructuradas.
Entrada/Pasos de ejecución: Bóveda difusa, la clave y la plantilla de minucias de consulta.
Resultado esperado: Mensaje de si corresponden las huellas dactilares al mismo rasgo biométrico.
Evaluación: Satisfactoria

Tabla 26. Caso de prueba de la funcionalidad: Comparar las plantillas en el dominio cifrado.

ANEXO#5: Estructura definida para las plantillas de minucias

La plantilla de minucias debe estar estructurada de la siguiente manera:

La primera línea contiene la cantidad de minucias obtenidas de la huella dactilar, la segunda línea refleja el punto $\boldsymbol{x}$, $\boldsymbol{y}$ donde se ubica el núcleo en el eje de coordenadas y de la tercera línea hasta el final de la plantilla se muestran las minucias cargadas, las cuales se representan como: $\{\boldsymbol{x}, \mathbf{y}, \boldsymbol{\theta}, \mathbf{t}\}$, donde $\boldsymbol{x}$, $\boldsymbol{y}$ es la posición en la imagen de la huella, $\boldsymbol{\theta}$ es el ángulo en radianes de dicha minucia y $\mathbf{t}$ es el tipo de minucia.

```
19
180 261
160 204 2 1
183 223 140 0
69 134 25 0
161 69 133 1
247 88 242 1
22 134 166 0
115 16 20 1
190 179 126 1
240 204 225 1
144 277 170 0
202 276 183 1
216 283 83 0
62 288 171 1
186 140 246 0
245 167 232 0
209 268 212 0
131 60 5 0
223 50 251 0
43 274 164 1
```

Figura 22. "Plantilla de minucias"

ANEXO#6: Resultados de las pruebas de efectividad.

Bóveda	Muestra	Comparación
M1	M2-1	FALSO
M1	M2-2	FALSO
M1	M2-3	FALSO
M1	M2-4	FALSO
M1	M2-5	FALSO
M1	M3-1	FALSO
M1	M3-2	FALSO
M1	M3-3	FALSO
M1	M3-4	FALSO
M1	M3-5	FALSO
M1	M4-1	FALSO
M1	M4-2	FALSO
M1	M4-3	FALSO
M1	M4-4	FALSO
M1	M4-5	FALSO
M1	M5-1	FALSO
M1	M5-2	FALSO
M1	M5-3	FALSO
M1	M5-4	FALSO
M1	M5-5	FALSO
M1	M6-1	FALSO
M1	M6-2	FALSO
M1	M6-3	FALSO
M1	M6-4	FALSO
M1	M6-5	FALSO
M1	M7-1	FALSO
M1	M7-2	FALSO
M1	M7-3	FALSO
M1	M7-4	FALSO
M1	M7-5	FALSO
M1	M8-1	FALSO
M1	M8-2	FALSO

Bóveda	Muestra	Comparación
M5	M2-1	FALSO
M5	M2-2	FALSO
M5	M2-3	FALSO
M5	M2-4	FALSO
M5	M2-5	FALSO
M5	M3-1	FALSO
M5	M3-2	FALSO
M5	M3-3	FALSO
M5	M3-4	FALSO
M5	M3-5	FALSO
M5	M4-1	FALSO
M5	M4-2	FALSO
M5	M4-3	FALSO
M5	M4-4	FALSO
M5	M4-5	FALSO
M5	M1-1	FALSO
M5	M1-2	FALSO
M5	M1-3	FALSO
M5	M1-4	FALSO
M5	M1-5	FALSO
M5	M6-1	FALSO
M5	M6-2	FALSO
M5	M6-3	FALSO
M5	M6-4	FALSO
M5	M6-5	FALSO
M5	M7-1	FALSO
M5	M7-2	FALSO
M5	M7-3	FALSO
M5	M7-4	FALSO
M5	M7-5	FALSO
M5	M8-1	FALSO
M5	M8-2	FALSO

M1	M8-3	FALSO
M1	M8-4	FALSO
M1	M8-5	FALSO
M2	M1-1	FALSO
M2	M1-2	FALSO
M2	M1-3	FALSO
M2	M1-4	FALSO
M2	M1-5	FALSO
M2	M3-1	FALSO
M2	M3-2	FALSO
M2	M3-3	FALSO
M2	M3-4	FALSO
M2	M3-5	FALSO
M2	M4-1	FALSO
M2	M4-2	FALSO
M2	M4-3	FALSO
M2	M4-4	FALSO
M2	M4-5	FALSO
M2	M5-1	FALSO
M2	M5-2	FALSO
M2	M5-3	FALSO
M2	M5-4	FALSO
M2	M5-5	FALSO
M2	M6-1	FALSO
M2	M6-2	FALSO
M2	M6-3	FALSO
M2	M6-4	FALSO
M2	M6-5	FALSO
M2	M7-1	FALSO
M2	M7-2	FALSO
M2	M7-3	FALSO
M2	M7-4	FALSO

M5	M8-3	FALSO
M5	M8-4	FALSO
M5	M8-5	FALSO
M6	M2-1	FALSO
M6	M2-2	FALSO
M6	M2-3	FALSO
M6	M2-4	FALSO
M6	M2-5	FALSO
M6	M3-1	FALSO
M6	M3-2	FALSO
M6	M3-3	FALSO
M6	M3-4	FALSO
M6	M3-5	FALSO
M6	M4-1	FALSO
M6	M4-2	FALSO
M6	M4-3	FALSO
M6	M4-4	FALSO
M6	M4-5	FALSO
M6	M5-1	FALSO
M6	M5-2	FALSO
M6	M5-3	FALSO
M6	M5-4	FALSO
M6	M5-5	FALSO
M6	M1-1	FALSO
M6	M1-2	FALSO
M6	M1-3	FALSO
M6	M1-4	FALSO
M6	M1-5	FALSO
M6	M7-1	FALSO
M6	M7-2	FALSO
M6	M7-3	FALSO
M6	M7-4	FALSO

M2	M7-5	FALSO
M2	M8-1	FALSO
M2	M8-2	FALSO
M2	M8-3	FALSO
M2	M8-4	FALSO
M2	M8-5	FALSO
M3	M2-1	FALSO
M3	M2-2	FALSO
M3	M2-3	FALSO
M3	M2-4	FALSO
M3	M2-5	FALSO
M3	M1-1	FALSO
M3	M1-2	FALSO
M3	M1-3	FALSO
M3	M1-4	FALSO
M3	M1-5	FALSO
M3	M4-1	FALSO
M3	M4-2	FALSO
M3	M4-3	FALSO
M3	M4-4	FALSO
M3	M4-5	FALSO
M3	M5-1	FALSO
M3	M5-2	FALSO
M3	M5-3	FALSO
M3	M5-4	FALSO
M3	M5-5	FALSO
M3	M6-1	FALSO
M3	M6-2	FALSO
M3	M6-3	FALSO
M3	M6-4	FALSO
M3	M6-5	FALSO
M3	M7-1	FALSO
M3	M7-2	FALSO
M3	M7-3	FALSO
M3	M7-4	FALSO
M3	M7-5	FALSO
M3	M8-1	FALSO
M3	M8-2	FALSO
M3	M8-3	FALSO
M3	M8-4	FALSO

M6	M7-5	FALSO
M6	M8-1	FALSO
M6	M8-2	FALSO
M6	M8-3	FALSO
M6	M8-4	FALSO
M6	M8-5	FALSO
M7	M2-1	FALSO
M7	M2-2	FALSO
M7	M2-3	FALSO
M7	M2-4	FALSO
M7	M2-5	FALSO
M7	M3-1	FALSO
M7	M3-2	FALSO
M7	M3-3	FALSO
M7	M3-4	FALSO
M7	M3-5	FALSO
M7	M4-1	FALSO
M7	M4-2	FALSO
M7	M4-3	FALSO
M7	M4-4	FALSO
M7	M4-5	FALSO
M7	M5-1	FALSO
M7	M5-2	FALSO
M7	M5-3	FALSO
M7	M5-4	FALSO
M7	M5-5	FALSO
M7	M6-1	FALSO
M7	M6-2	FALSO
M7	M6-3	FALSO
M7	M6-4	FALSO
M7	M6-5	FALSO
M7	M1-1	FALSO
M7	M1-2	FALSO
M7	M1-3	FALSO
M7	M1-4	FALSO
M7	M1-5	FALSO
M7	M8-1	FALSO
M7	M8-2	FALSO
M7	M8-3	FALSO
M7	M8-4	FALSO

M3	M8-5	FALSO
M4	M1-1	FALSO
M4	M1-2	FALSO
M4	M1-3	FALSO
M4	M1-4	FALSO
M4	M1-5	FALSO
M4	M3-1	FALSO
M4	M3-2	FALSO
M4	M3-3	FALSO
M4	M3-4	FALSO
M4	M3-5	FALSO
M4	M2-1	FALSO
M4	M2-2	FALSO
M4	M2-3	FALSO
M4	M2-4	FALSO
M4	M2-5	FALSO
M4	M5-1	FALSO
M4	M5-2	FALSO
M4	M5-3	FALSO
M4	M5-4	FALSO
M4	M5-5	FALSO
M4	M6-1	FALSO
M4	M6-2	FALSO
M4	M6-3	FALSO
M4	M6-4	FALSO
M4	M6-5	FALSO
M4	M7-1	FALSO
M4	M7-2	FALSO
M4	M7-3	FALSO
M4	M7-4	FALSO
M4	M7-5	FALSO
M4	M8-1	FALSO
M4	M8-2	FALSO
M4	M8-3	FALSO
M4	M8-4	FALSO
M4	M8-5	FALSO
M7	M8-5	FALSO
M8	M2-1	FALSO
M8	M2-2	FALSO
M8	M2-3	FALSO
M8	M2-4	FALSO
M8	M2-5	FALSO
M8	M3-1	FALSO
M8	M3-2	FALSO
M8	M3-3	FALSO
M8	M3-4	FALSO
M8	M3-5	FALSO
M8	M4-1	FALSO
M8	M4-2	FALSO
M8	M4-3	FALSO
M8	M4-4	FALSO
M8	M4-5	FALSO
M8	M5-1	FALSO
M8	M5-2	FALSO
M8	M5-3	FALSO
M8	M5-4	FALSO
M8	M5-5	FALSO
M8	M6-1	FALSO
M8	M6-2	FALSO
M8	M6-3	FALSO
M8	M6-4	FALSO
M8	M6-5	FALSO
M8	M7-1	FALSO
M8	M7-2	FALSO
M8	M7-3	FALSO
M8	M7-4	FALSO
M8	M7-5	FALSO
M8	M1-1	FALSO
M8	M1-2	FALSO
M8	M1-3	FALSO
M8	M1-4	FALSO
M8	M1-5	FALSO
FMR		0

Figura 23. "Pruebas de falsos aceptados"

Bóveda	Muestra	Comparación
M1	M1-1	FALSO
M1	M1-2	FALSO
M1	M1-3	VERDADERO
M1	M1-4	FALSO
M1	M1-5	VERDADERO
M2	M2-1	VERDADERO
M2	M2-2	VERDADERO
M2	M2-3	FALSO
M2	M2-4	VERDADERO
M2	M2-5	VERDADERO
M3	M3-1	FALSO
M3	M3-2	FALSO
M3	M3-3	VERDADERO
M3	M3-4	VERDADERO
M3	M3-5	FALSO
M4	M4-1	VERDADERO
M4	M4-2	FALSO
M4	M4-3	VERDADERO
M4	M4-4	VERDADERO
M4	M4-5	VERDADERO
M5	M5-1	FALSO
M5	M5-2	FALSO
M5	M5-3	VERDADERO
M5	M5-4	VERDADERO
M5	M5-5	VERDADERO
M6	M6-1	VERDADERO
M6	M6-2	VERDADERO
M6	M6-3	VERDADERO
M6	M6-4	VERDADERO
M6	M6-5	VERDADERO
M7	M7-1	FALSO
M7	M7-2	FALSO
M7	M7-3	VERDADERO
M7	M7-4	VERDADERO
M7	M7-5	FALSO
M8	M8-1	FALSO
M8	M8-2	VERDADERO
M8	M8-3	VERDADERO
M8	M8-4	FALSO
M8	M8-5	FALSO
FNMR		0,4

Figura 24. "Pruebas de falso rechazo"