UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMATICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACION

“DIESÑO E IMPLEMENTACION DE UN ESQUEMA DE ENCRIPTACION Y FIRMAS BASADO EN IDENTIDAD PARA DISPOSITIVOS BUG”

MEMORIA PARA OPTAR AL TITULO DE INGENIERO CIVIL EN COMPUTACION

MARIA FRANCISCA MORENO VILICICH

SANTIAGO DE CHILE

UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMATICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACION

"DIESÑO E IMPLEMENTACION DE UN ESQUEMA DE ENCRIPTACION Y FIRMAS BASADO EN IDENTIDAD PARA DISPOSITIVOS BUG"

MEMORIA PARA OPTAR AL TITULO DE INGENIERO CIVIL EN COMPUTACION

MARIA FRANCISCA MORENO VILICICH

PROFESOR GUÍA:
ALEJANDRO HEVIA ANGULO

MIEMBROS DE LA COMISION
PABLO BARCELÓ BAEZA
VALERIA HERSKOVIC MAIDA

SANTIAGO DE CHILE
DICIEMBRE 2010

"-Es muy peligroso, Frodo, cruzar la puerta- decía Bilbo –vas hacia el camino y si no cuidas tus pasos no sabes hacia dónde te arrastrarán-"

Frodo Bolsón, "La Comunidad del Anillo".
J.R.R. Tolkien

Agradecimientos

Gracias a mi familia, por el constante apoyo e impulso en todos estos años de esfuerzo. Llegué a donde estoy porque ustedes fueron mi puntal. Gracias por la infinita paciencia, por las conversaciones, por ser mi familia siempre, incluso cuando yo no lo fui.

Gracias a mi profesor guía, por la fe y la confianza en el proyecto, por los consejos, apoyo y por el tiempo dedicado. Fue un gran placer ser su alumna.

Gracias al profesor Eduardo Vera, por introducirme al proyecto, por darme un lugar donde no estaba seguro que había uno y por darme la línea de partida en mi carrera laboral

Gracias a mis amigos, por la infinita paciencia de mis ausencias, por el ánimo y el apoyo para levantarme con cada caída. Gracias por estar ahí a pesar de que yo no siempre pude estar

Y por último, gracias a Gonzalo. Tú fuiste quien mantuvo vivos mis sueños, cuando el mundo me decía que aterrizara y no estaría donde estoy si no fuera por eso. Gracias por estar en cada etapa, en cada semestre, gracias por estar en mi pasado, presente y futuro.

Resumen Ejecutivo

Históricamente, la autentificación de mensajes se ha manejado a través de firmas digitales basadas en certificados, usualmente emitidos por una entidad neutral a la comunicación y de confianza de todas las partes. Sin embargo, esta solución no siempre es posible de implementar, debido a factores como los costos, el tiempo limitado de vigencia de los certificados, la necesidad de depositar la confianza en un tercero, y por último el poder, usualmente limitado, de procesamiento de las máquinas. Además los certificados usualmente son muy pesados, y deben adjuntarse a cada mensaje enviado para mostrar su autenticidad, lo cual dificulta su implementación en dispositivos con poca capacidad de almacenamiento.

La Criptografía basada en identidad (o Identity Based Cryptography, IBC) es una alternativa al escenario presentado en sistemas que desean autentificar mensajes, pero no quieren (o no pueden) depender de certificados. Esto se logra usando la identidad de la persona (o del dispositivo) como semilla para la generación de claves por parte de un servidor maestro dentro del mismo sistema. De esta forma, las firmas son mucho más ligeras que los certificados, agilizando las comunicaciones. Además, como las identidades son públicas, cualquier miembro de la red puede verificar la autenticidad de un mensaje sin necesidad de hacer consultas online al ente central.

Por otro lado, las lecturas biométricas son la forma más usada de comprobar la identidad de una persona, ya sea a través de las huellas digitales, reconocimiento facial, lectura de iris o incluso análisis de forma de las orejas. Esto debido a que los rasgos mencionados son únicos para la persona, e incluso los dispositivos actuales permiten reconocer rasgos latentes, es decir que no poseen señales biológicas, de "vivos" y así prevenir el uso de guantes, máscaras, etc.

El tema de esta tesis es desarrollar una alternativa de sistema de seguridad para dispositivos de procesamiento limitado. El sistema utiliza firma de mensajes basada en identidad, pero usando las huellas digitales tanto para enlazar un usuario específico a un dispositivo, como para la protección de los datos privados del usuario. El sistema debe poder extenderse para poder incorporar encriptación basada en identidad también.

La primera parte de este trabajo consiste en un estudio de las componentes a usar, tanto de hardware como de software, incluyendo los conceptos básicos necesarios para su diseño y operación, así como los estándares que siguen. La segunda parte corresponde a un estudio y descripción detallada de las componentes del proyecto y de la implementación realizada, para terminar con posibles extensiones.

El sistema se probará en dispositivos BUG, un tipo de hardware usado en el diseño y fabricación de prototipos modulares.

Tabla de contenido

Tabla de Contenido: Ilustraciones

Capítulo 1: Introducción

1.1. Reseña Histórica, Contexto y Motivación

El concepto de criptografía basada en identidad apareció en 1984 acuñado por Adi Shamir [9]. La idea original (y que se mantiene hasta hoy en día) era diseñar e implementar un protocolo de criptografía de clave pública en la cual no fuera necesario un intercambio de claves entre los usuarios, usando una tercera entidad que entregara a cada nuevo miembro toda la información necesaria para poder operar de forma independiente, es decir, que pueda encriptar, desencriptar, firmar y verificar cualquier mensaje, sin importar quien esté al otro lado de la comunicación. El esquema original contaba con la implementación de los centros de generaciones de claves y la información que entregaban los usuarios era cualquier combinación de datos que los identificara de forma única, la cual era procesada por los centros y devuelta a los usuarios como tarjetas inteligentes (smartcards) o dispositivos con baja capacidad de procesamiento.

Sin embargo, por novedoso que fuera en el momento, Shamir sólo logró definir la teoría para implementar un protocolo de autentificación con este modelo, dejando el tema de la encriptación como problema abierto hasta el año 2001, cuando en forma independiente Boneh y Franklin [2], y también Cocks [4] presentaron sus construcciones en CRYPTO'01 e IMA '01 respectivamente, transformándose rápidamente en los modelos más usados y estudiados para resolver el problema.

Independiente del protocolo para el que sea usado, el manejo de identidades se beneficia de sobremanera con el uso de lecturas biométricas de cualquier tipo, ya que permiten un menaje de claves que el usuario no necesita recordar. Sin embargo, tales lecturas son imprecisas, por lo que requieren de un algoritmo especial para que puedan ser usadas en el contexto criptográfico. Tales algoritmos, llamadas "extractores imprecisos" (fuzzy extractors en inglés -y el nombre que será usado en este trabajo) permiten la obtención de una cadena única de caracteres a partir de datos imprecisos, mientras sean lo suficientemente similares a un patrón de referencia, y pueden ser usados para comprobar la identidad de una persona a partir de una lectura biométrica disponible a través de los hitos respectivos (minucias en el caso de las huellas digitales).

Situándose en el contexto de las redes ad-hoc, específicamente las formadas por dispositivos BUGs que son elementos de hardware modular diseñado para la fabricación de prototipos. Estas implementaciones poseen gran potencial ya que permiten dar soluciones a problemas, por ejemplo, que el estado de salud de un paciente sea comunicado a su médico, aún cuando ninguno de los dos se encuentre en el hospital o consulta. Con la ayuda de un módulo GPS se podría lograr el monitoreo autentificado de obreros, o miembros de la policía en operaciones riesgosas.

En todas estas situaciones las redes ad-hoc móviles son útiles debido a que toleran ambientes con poca conectividad, ya sea armando una red privada y comunicándose entre ellos, hasta que alguno de los miembros encuentra un acceso a internet y se transforme en gateway para la red. Sin embargo dada la movilidad de la conexión con el exterior, es imperante asegurar de alguna forma la confidencialidad y autenticidad de los mensajes, a través de autentificación y encriptación para poder confirmar la identidad de quien mande mensajes y así evitar que información sea inyectada por alguien externo a la red.

Tradicionalmente esto se logra adjuntando un certificado al mensaje ya firmado, mostrando que una entidad externa asegura la identidad de quien manda el mismo. Sin embargo, este método tiene las

desventajas de que requiere tiempo y una conexión permanente con la entidad certificadora, ya que se debe obtener el certificado cada vez que se desea enviar un mensaje. Además los certificados expiran y son costosos de transmitir, por lo que este procedimiento no es considerado conveniente para una red ad-hoc.

La alternativa es utilizar criptografía basada en identidad, usando una lectura biométrica como identidad de la persona. Esto remueve la necesidad de los certificados, pero no la existencia de la entidad certificadora, que ahora se focaliza en la generación de claves de los usuarios.

Este trabajo tiene el objetivo de presentar una alternativa "híbrida" al problema, usando autentificación basada en identidad con la identidad del dispositivo (IPv6, MAC, etc.) para autentificar los mensajes, logrando que la clave privada sólo sea generada una vez al configurarse y protegiendo esta clave mediante un sistema de criptografía simétrico (AES), cuya clave es derivada de una lectura biométrica a través de un fuzzy extractor, logrando así unir la identidad de un usuario a la de un dispositivo en particular, ya que cada implementación estaría funcional para una sola persona. Conceptualmente la idea es emular el llamado "tamper proof hardware" a través de software, ya que cualquier intento de obtener la información privada de forma ilegal resulta en la pérdida de la información porque la única información que se guarda sin encriptar es información pública y no suficiente por sí misma para desencriptar las claves. Desencriptar la información privada requiere la información de la huella digital, la cual es proporcionada por el usuario en el momento de autentificarse y borrada luego de ello.

1.2. Objetivos y plan de trabajo

1.2.1. Objetivo General

El objetivo de este trabajo de memoria es diseñar e implementar un sistema que permita asegurar la autenticidad para las comunicaciones dentro de las redes ad-hoc formadas por dispositivos BUG, además de dejar las bases sentadas para incorporar una implementación de un sistema de encriptación en el futuro.

1.2.2. Objetivos Específicos

1. Realizar un estudio del arte de la criptografía basada en identidad, tanto en la encriptación de mensajes, como en la autentificación de usuarios.

2. Diseñar un protocolo de autentificación que permita ser implementado en redes ad-hoc sin la necesidad de la conexión permanente con una tercera entidad que genere las claves necesarias. Para este fin se usará un modelo que combine autentificación basada en identidad con criptografía simétrica fuzzy para la protección de los datos privados y a la vez permitir demostrar la identidad del usuario que maneja el dispositivo.

3. Desarrollo de una API dedicada a la seguridad en comunicación de los datos, basada en los resultados de los dos puntos anteriores, enfocándose en redes de dispositivos electrónicos de capacidad limitada y que poseen alguna interfaz para permitir a los usuarios demostrar su identidad, ya sean lectores biométricos o algún medio de ingreso de datos para responder preguntas o escribir una contraseña. Para este punto, se contempla el uso de dispositivos BUG con la interfaz apropiada.

4. Desarrollo de una aplicación a través de la cual se pueda ver el trabajo de la API desarrollada, o mostrar el desarrollo incluido en una aplicación ya existente.

5. Estudiar mecanismos y técnicas de procesamiento y comparación de huellas digitales.

1.3. Plan de trabajo

En este trabajo se busca comprender los componentes fundamentales de un sistema de autentificación de identidad basado en biometría, y sus aplicaciones para poder firmar mensajes de forma segura a través del sistema de IBC. Para lograr esto se estudiarán y evaluarán diversas propuestas que existen en la actualidad, sobre todo en los distintos esquemas de firmas.

Para llevar a cabo la evaluación se comenzará evaluando la propuesta original de Shamir para Identity Based Signatures. Tomando esta propuesta como base, se evaluará cómo ha ido evolucionando en el tiempo a través de los trabajos de distintas personas.

Teniendo ya una comprensión amplia del tema, se procederá a evaluar una propuesta más apropiada para una implementación real, como es el trabajo de Paterson [7] en firmas, y Dodis, Reyzin y Smith en la parte de fuzzy extractors [5], que a su vez son los trabajos más recientes que han sido publicados en conferencias de nivel internacional.

Una vez realizada el estudio de alternativas, se pretende diseñar y lograr la implementación final de una API hecha para un ambiente general combinando todas las partes del sistema.

Con la API funcionando completamente, sigue una fase de adaptación para los dispositivos BUG, teniendo en cuenta sus limitaciones y características. Eso se prueba a través de una aplicación sencilla en la cual un BUG se comunique con un computador, simulando el paso final de una red ad-hoc.

Capítulo 2: Conceptos Básicos

2.1. Notaciones y teoría matemática.

2.1.1. Notaciones

El conjunto de los números enteros se denominará $\mathbb{Z}$ mientras que $\mathbb{Z}_N^*$ denominará el conjunto de los enteros positivos menores a N primos relativos de N. Letras como $\mathbb{F}, \mathbb{G}$ denominan grupos y $\mathbb{R}_+$ denomina el conjunto de los reales positivos

Cuando se tiene una distribución de probabilidades con esperanza $\mathcal{E}$, se denomina $\mathcal{E}_{b \leftarrow B}$ al valor esperado de la probabilidad cuando la variable B tiene valor b y se denomina como la predictibilidad de A $max_a Pr[A = a]$ la máxima probabilidad de que una variable aleatoria A tome un valor fijo a.

2.1.2. Emparejamientos Bilineales

Para este trabajo no es necesario un entendimiento cabal de Emparejamientos Bilineales, lo cual incluiría teoría de curvas elípticas. Más bien, entregaremos algunas definiciones básicas suficientes para comprender lo desarrollado más adelante.

Sean $\mathbb{F}, \mathbb{G}$ dos grupos de tamaño q primo. $\mathbb{G}$ está descrito en términos de una operacion aditiva y formado por puntos de una curva elíptica, mientras que $\mathbb{F}$ está descrito en términos de una operacion multiplicativa y formado por un subconjunto de un campo finito. Un par bilineal es un mapeo $\hat{e}: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{F}$ con las siguientes propiedades:

1. Bilinearidad: Si $R_1, R_2 \in \mathbb{G} \wedge a, b \in \mathbb{Z}_q^*$ entonces $\hat{e}(aR_1, bR_2) = \hat{e}\left(R_1, R_2\right)^{ab}$
2. No degenerativo: $\hat{e}$ no transforma todos los puntos de $\mathbb{G} \times \mathbb{G}$ en un único punto de $\mathbb{F}$
3. Eficiencia: el cálculo de $\hat{e}$ es eficiente para cualquier elemento del dominio.

2.1.3. Problema Diffie-Hellman Bilinear

El problema de Diffie-Hellman original se basa en que, dado un valor g (el generador de un grupo) y x, y enteros aleatorios, si se entrega g^x, g^y calcular g^{xy} es computacionalmente dificil.

La variante bilinear de este problema (Bilinear Diffie-Hellman o BDH) se basa en ese mismo principio. Si se tiene un grupo $\mathbb{G}$ de tamaño q primo, con un mapa $\hat{e}$, y a, b, c elementos al azar de $\mathbb{Z}_q^*$ si se entrega la tupla $(\mathbb{G}, q, \hat{e}, P, aP, bP, cP)$, con P en $\mathbb{G}$ calcular $\hat{e}(P, P)^{abc}$ es computacionalmente dificil.

2.1.4. Espacios Métricos

Para un conjunto M, el espacio métrico dado por este es el conjunto más una función de distancia, que cumple las reglas de simetría y desigualdad triangular.

$$dis: M \times M \rightarrow \mathbb{R}^{+} = [0,\infty)$$
$$dis(x,y) = dis(y,x)$$
$$dis(x,y) \leq dis(x,z) + dis(z,y)$$

Para este trabajo, M está formado por todos los subconjuntos posibles de un universo dado, mientras que la función de distancia corresponde a la diferencia simétrica entre dos elementos, esto es $dis(\omega,\omega') \stackrel{\text{def}}{=} |\omega \vartriangle \omega'|$

2.1.5. Entropía mínima y entropía mínima promedio.

Para una variable aleatoria A, se define su entropía mínima como menos el logaritmo de su predictibilidad, es decir.

$$\mathbf{H}_{\infty}(A) = -\log(max_a Pr[A = a])$$

Por otro lado, si se tienen dos variables relacionadas entre sí, la predictibilidad de una depende de la otra, o sea

$$max_a Pr[A = a \mid B = b]$$

Por lo tanto, cuando existen dos variables, la entropía mínima pasa a ser entropía mínima promedio, definida por:

$$\widetilde{\mathbf{H}}_{\infty}(A|B) \stackrel{\text{def}}{=} -\log\left(E_{b\leftarrow B}\left[max_a Pr[A = a|B = b]\right]\right) = -\log\left(E_{b\leftarrow B}\left[2^{-\mathbf{H}_{\infty}(A|B=b)}\right]\right)$$

Esta teoría, usada en los fuzzy extractors, indica la cantidad de bits aleatorios que se pueden obtener con una distribución dada, lo que da una cota superior de la cantidad de información que se puede extraer para una entrada de datos determinada.

2.2. Redes Ad-hoc

Son redes formadas por una cantidad variable de dispositivos, fijos y/o móviles que cuentan con comunicación de corto alcance, por lo que la información es distribuida haciéndola pasar a través de los miembros de la red. Además, frecuentemente en estas redes no existe una comunicación con el exterior (internet). Cuando alguno de los miembros de la red logra abrir una conexión con el exterior, como por ejemplo entrando a una zona cubierta por un punto de acceso, este toma el papel de gateway. Este rol no es fijo ni único, por lo que es necesario que, dentro de la red, los miembros repartan la información a la mayor cantidad de dispositivos conectados posibles, para asegurar su salida en caso de que un gateway aparezca. Un uso común para estas redes es de sensores con sus respectivos lectores (mezcla de dispositivos fijos y móviles).

2.3. Dispositivos BUG

Los BUG [10] son dispositivos para diseñar prototipos de forma modular. Cuentan con una base, que lleva un sistema operativo Linux y varios módulos que son conectables a la base para ocupar sus funcionalidades. Entre los módulos más usados se encuentran el acelerómetro, el reproductor de sonidos, GPS y sensor infrarrojo. Distintas aplicaciones pueden crearse dependiendo de los módulos conectados a la base y éstas son programables a través de Dragonfly, el plug-in para IDE Eclipse, el cual permite transferir las aplicaciones programadas al BUG mismo mediante una interfaz USB y a través de Wi-fi. Los BUGs tienen conectividad a través de protocolos Wi-fi, Bluetooth y ZigBee, todos ellos integrados en la base del dispositivo. Versiones anteriores implementan la conectividad en un módulo aparte.

Además, a través de la interfaz USB, es posible conectar otros dispositivos, como en este caso es un lector de huellas digitales USB, marca Futronic.

2.4. Criptografía Basada en Identidad

Una alternativa dentro de los sistemas de criptografía de clave pública, consiste en usar la identidad de un usuario como semilla para la creación del par de claves que usará el sistema. Estas identidades son públicas y de libre acceso, pero las claves privadas asociadas a ellas son generadas por una entidad central de la confianza de todos a la cual acceden los usuarios. En la Ilustración 1 se detalla el proceso de encriptación de un mensaje por identidad.

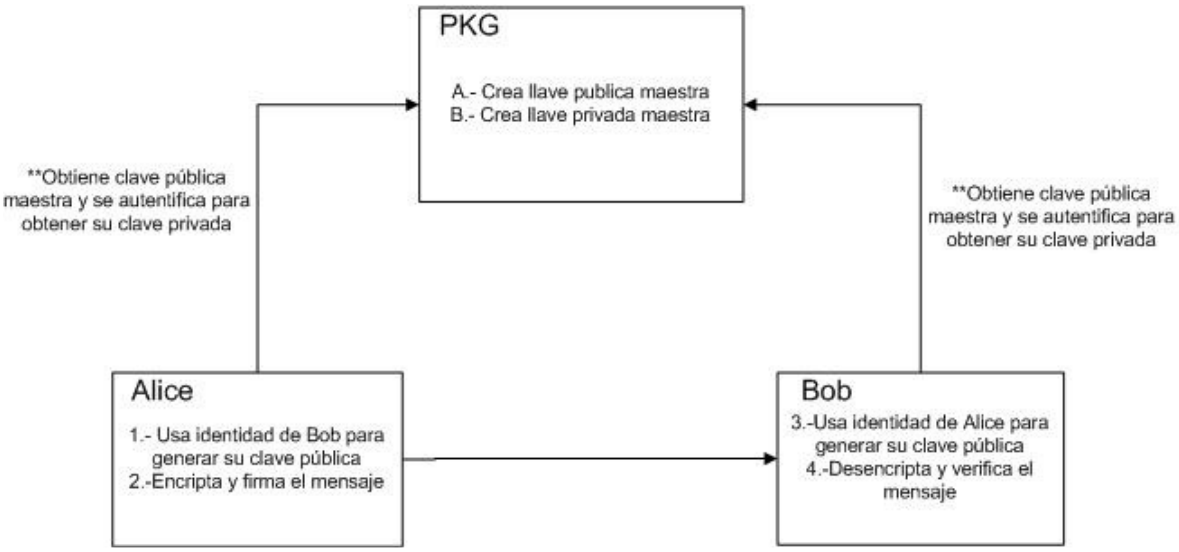

Ilustración 1: Esquema básico de un sistema de criptografía basado en identidades

Este mismo protocolo puede ser usado para la generación de firmas y su verificación, tomando en cuenta que para crear la firma se necesita la clave privada y para verificarla se usa la clave pública. Dado que ambos algoritmos (encriptación y firma) usan las mismas herramientas, es común implementarlos juntos.

2.5. Criptografía Fuzzy

Corresponde a una implementación de un protocolo de clave simétrica complementado con una estructura criptográfica que permite obtener una cadena fija a partir de datos imprecisos, mientras éstos sean lo suficientemente similares a un dato de referencia, llamada fuzzy extractor[5]. Un fuzzy extractor es un algoritmo que, tomando como entrada información potencialmente imprecisa (como una huella digital), destila de ella un identificador único para la huella. Es éste identificador el que es almacenado ya sea para crear firmas, claves de encriptación, claves de acceso entre otros.

Un ejemplo claro del uso de los fuzzy extractors es la transformación de una lectura de huella digital, usualmente una imagen o una cadena de caracteres, en otra cadena que refleje la identidad de la persona. Las lecturas biométricas siempre son imprecisas (posición del dedo, estado del escáner, cicatrices, suciedad, etc.), pero este proceso permite derivar la identidad de la persona a partir de la huella, siempre y cuando la lectura sea lo suficientemente parecida a la primera lectura tomada en el sistema, cuando se ingresó el usuario. Cuan parecidas deben ser dos lecturas está dado por un umbral en el fuzzy extractor, que puede implementar varias alternativas de criterios, siendo los más conocidos la Distancia de Hamming, la diferencia entre conjuntos (Set Difference) y distancia de edición (Edit Distance).

2.6. Procesamiento e identificación de huellas digitales

Cada huella digital es esencialmente única, ni siquiera los gemelos tienen huellas idénticas. De cada huella se extrae una serie de hitos dentro de la huella llamados minucias [12], los cuales permiten hoy en día identificar a una persona[1].

Las minucias pueden ser muchos tipos: bifurcaciones, vueltas en U, deltas, islas, etc., etc. La ilustración 2 muestra algunas minucias.

Ilustración 2: Distintos tipos de minucia en una huella digital.

[1] Cabe aclarar que es posible la existencia de colisiones en los patrones de minucias, si es que el universo de huellas digitales se vuelve muy grande (del orden de billones de personas)

Esta combinación de hitos, así como su posición y ángulo forman el patrón de la huella, que es lo que se usa a la hora de identificar a la persona. La alternativa es comparar la imagen completa, pero eso toma mucho tiempo y recursos, aunque así se hacía en el principio.

El proceso de extracción de las minucias desde una imagen de la huella consta de varias fases, las cuales permiten obtener una imagen pura en blanco y negro (como si hubiera sido dibujada con un lápiz fino) la cual es recorrida buscando los patrones de las minucias. Cada vez que una es encontrada, se toma su posición con respecto al origen del sistema así como el ángulo con respecto al eje horizontal. Junto a estos datos se almacenan varios más, como la calidad de las minucias, sin embargo estos últimos datos son simplemente para los softwares que los usan y no tienen mucha importancia para nuestro trabajo.

El proceso de identificación a través de las huellas consiste primero en lo que se llama enrolamiento, que es obtener una "huella patrón" de la persona usualmente cuando esta se registra en el sistema. A esta huella se le extraen las minucias y éstas son almacenadas en una base de datos. Cuando la persona quiere identificarse, se obtiene una huella en el momento y las minucias extraídas de estas son comparadas con las obtenidas en el momento del enrolamiento. Si la comparación pasa cierto umbral de porcentaje, la huella es considerada correcta y la persona es autentificada.

Para el almacenamiento de esta información y permitir que las empresas puedan ocupar distintos lectores de forma independiente al software, existen varios estándares entre ellos el ISO 19794-2 [14] cuya última revisión fue este año (2010).

Capítulo 3: Estado del Arte

3.1. Criptografía Basada en Identidad

Este tipo de esquemas criptográficos tienen varias raíces en la teoría matemática, ya sea de "Emparejamientos bilineales" ("Bilinear Pairing") [2] o en residuos cuadráticos [4]. Para este trabajo se estudió la implementación del trabajo de Boneh y Franklin basado en la teoría de Emparejamientos Bilineales, en específico el de Weil [2] para el tema de encriptación, mientras que para el trabajo con firmas se estudió el trabajo de Paterson [7] que usa las mismas bases que el esquema de encriptación, lo que lo hace muy conveniente a la hora de una única implementación.

Como la mayoría los trabajos de IBC, el esquema de encriptación de Boneh y Franklin se basa en el problema de Diffie-Hellman Bilinear y consta de 4 algoritmos para lograr la encriptación.

Setup: Es el algoritmo que permite a la entidad central (PKG) obtener las claves maestras y los parámetros públicos del sistema, de tal forma de que los miembros puedan operar.

1. Sea P un generador de $\mathbb{G}$ y $s \in \mathbb{Z}_q^*$ aleatorio. PKG fija $P_{pub} = sP$.
2. PKG escoge dos funciones de hash $H_1: \{0,1\}^* \to \mathbb{G}^*, H_2: \mathbb{F} \to \{0,1\}^n$ donde n es el largo de un texto plano.
3. La clave maestra es s y los parámetros públicos son $mpk = \langle P, P_{pub}, H_1, H_2 \rangle$.

Creación de claves: Cuando un usuario se registra, éste contacta a la entidad central, PKG y entregando su identidad, se le genera una clave privada de la siguiente forma:

1. Sea $id \in \{0,1\}^*$ una identidad. PKG Calcula $Q_{id} = H_1(id)$ y le devuelve la clave privada $d_{id} = sQ_{id}$.

Encriptación: Se ejecuta cada vez que un usuario quiere encriptar un mensaje. Funciona sin contacto con la entidad central.

1. Sea $\mathcal{M} \in \{0,1\}^n$ el texto plano a cifrar y sea i $id' \in \{0,1\}^*$ la identidad del destinatario.
2. Calcular $Q_{id'} = H_1(id')$ y escoger $r \in \mathbb{Z}_q^*$ al azar.
3. El texto cifrado es $C = \langle rP, \mathcal{M} \oplus H_2\left(\hat{e}(Q_{id}, P_{pub})^r\right) \rangle$

Desencriptación: El proceso inverso al algoritmo de encriptación

1. Sea $C = \langle U, V \rangle$ definido en el algoritmo anterior y d_{id} la clave privada del destinatario.
2. El texto plano es $\mathcal{M} = V \oplus H_2\left(\hat{e}(d_{id}, U)\right)$

Este esquema de encriptación es seguro contra ataques de texto plano escogido, es decir, ind-ibe-cpa secure y modificando el esquema según el trabajo de Fujisaki y Okamoto [1] es posible hacerlo seguro contra ataques de texto cifrado escogido (ind-ibe-cca secure). En términos simples, que un esquema sea ind-cpa secure significa que un adversario (alguien que quiere saber la clave del esquema, o el contenido de los mensajes encriptados) no puede distinguir entre textos cifrados, aun cuando él puede escoger los textos planos a encriptar. Por otro lado el término ind-cca secure significa que no se

puede distinguir entre textos cifrados, aun cuando el adversario puede escoger los textos planos y los cifrados, es decir que puede pedirle al esquema tanto encriptaciones como desencriptaciones [6].

Ya se mencionó que el esquema de autentificación usa las mismas bases que el esquema de encriptación. Además el esquema utiliza lo siguiente:

1. Se conoce un punto $P \in \mathbb{G}_1$ tal que $\hat{e}(P,P) \neq 1_{\mathbb{G}_2}$, donde $1_{\mathbb{G}_2}$ corresponde al neutro multiplicativo en $\mathbb{G}_2$
2. Se conocen las siguientes funciones de hash:
   a. $H_1: \{0,1\}^* \rightarrow \mathbb{G}$. Esta función puede ser la misma que la usada para encriptación y de hecho se recomienda que sea así.
   b. $H_2: \{0,1\}^* \rightarrow \mathbb{Z}_q$
   c. $H_1: \mathbb{G} \rightarrow \mathbb{Z}_q$
3. La clave privada del usuario con identidad id es d_{id}

Los algoritmos de este esquema son cuatro: Setup, Keygen, Sign y Verify, donde los dos primeros son los mismos que los usados por la entidad central para el esquema de encriptación. Los algoritmos Sign y Verify son como sigue:

Sign: El algoritmo de firma de mensajes

1. Se escoge $k \in \mathbb{Z}_q^*$ aleatorio
2. Se calcula $R = sP$ y $S = k^{-1}(H_2(M) * P + H_3(R) * d_{id})$
3. La firma es (R,S)

Verify: El algoritmo para verificar si la firma que acompaña a un mensaje corresponde a ese mensaje y ese usuario

1. Dada una firma (U,V) calcular $\hat{e}(U,V)$
2. Comparar el resultado con $\hat{e}(P,P)^{H_2(M)} * \hat{e}(P_{pub}, Q_{id})^{H_3(R)}$
3. Si ambos valores coinciden, la firma es válida. Sino el mensaje debe ser rechazado.

Este esquema de firmas es seguro contra ataques de mensajes escogidos, es decir uf-cma secure. Esto quiere decir que el atacante no puede crear la firma para una combinación de texto e identidad determinados por él, sin importar cuantas firmas válidas haya obtenido del usuario [6].

3.2. Fuzzy Encryption

Hoy en día, esta área está enfocada hacia el uso de identidades y lecturas biométricas. La principal aplicación para esta área son los Fuzzy Extractors, cuyo papel es extraer información desde las lecturas biométricas de tal forma de poder formar una cadena única de caracteres para cada instancia biométrica guardada en el sistema. Dado que este tipo de lecturas son imprecisas, se requiere un umbral de error de forma que lecturas similares, pero no idénticas sean tomadas como correctas y los fuzzy extractors logran eso tomando aleatoriamente información de la lectura, que al final sólo se traduce a escoger un número determinado de bytes al azar. Esta información permite más tarde reconstruir el dato (lectura) completo. Por ejemplo, si se toma como lectura biométrica una huella digital el fuzzy extractor tomará una cantidad fija de bytes de la lectura y los usará como semilla para crear la clave del sistema criptográfico simétrico que se está usando. Importantes son los trabajos de Sahai [8], Dodis, Reyzen y Smith [5] y Boyen, todos publicados el año 2004.

La implementación de los fuzzy extractors hace uso de un algoritmo llamado "secure sketch" para su construcción, aparte de algoritmos de corrección de errores. Todos estos elementos serán explicados con más detalle en las secciones siguientes.

Dependiendo del universo donde se trabaje, existen distintas variantes de fuzzy extractors. Debido a que en este caso cada dispositivo es usado por una sola persona, el universo a reconocer es consideradamente pequeño (un elemento), por lo que se decidió trabajar con la variante de secure sketch que usa diferencia de conjuntos en espacios no transitivos, denominada pinsketch[5].

Secure Sketch: Es un par de funciones, "sketch" (SS) y "recuperar" (Rec) que cumplen las siguientes propiedades:

1. La función SS, para una entrada determinada $w \in \mathcal{M}$ retorna una cadena de bits $s \in \{0,1\}^*$. Denotaremos s como $SS(w)$

2. Correctitud: La función Rec toma un elemento $w' \in \mathcal{M}$ y una cadena $s \in \{0,1\}^*$. Si $dis(w, w') \leq t$ entonces $Rec\left(w', SS(w)\right) = w$. Esta propiedad significa que, si un dato es lo suficientemente cercano a otro, entonces la función Rec puede recuperar el dato original con la ayuda de la cadena que se obtuvo con la función SS, asumiendo que dos datos son cercanos si son lo suficientemente similares.

3. Seguridad: La probabilidad de que un adversario pueda recuperar la distribución W a partir de la observación de $SS(w)$ es como máximo $2^{-\tilde{m}}$ donde $\tilde{m}$ es un parámetro del secure sketch. Esto significa que $\tilde{\mathbf{H}}_\infty\left(W \middle| SS(W)\right) \geq \tilde{m}$.

Fuzzy Extractor: Corresponde nuevamente a un par de funciones "Generar" (Gen) y "Reproducir" (Rep) con las siguientes propiedades:

1. La función Gen toma un elemento $w \in \mathcal{M}$ y entrega una cadena $r \in \{0,1\}^l$, correspondiente a la información aleatoria y una cadena $p \in \{0,1\}^*$ que sirve de ayuda para la posterior reconstrucción de r a partir de otra entrada w' similar. Se puede ver que la cadena r tiene un largo fijo independiente del tamaño de la entrada

2. Correctitud: Asumiendo que la función Rep toma por entrada un elemento $w' \in \mathcal{M}$ y una cadena $p \in \{0,1\}^*$ se cumple que si se da: $dis(w, w') \leq t$ y que $(r, p) \leftarrow Gen(w)$, entonces $Rec(w', p) = w$

3. Seguridad: Si $(r, p) \leftarrow Gen(w)$ con w una distribución sobre $\mathcal{M}$ con una entropía mínima m, entonces aun cuando un adversario tenga p, r le parecerá cuasi uniforme.

Estas propiedades quieren decir que, usando la cadena p es posible reconstruir r a partir de cualquier lectura que sea similar a la original, sin la necesidad de que p sea secreto, porque el que sea de conocimiento público o no, no afecta la cantidad de información que se pueda obtener de r.

Ahora, aunque ambos algoritmos son independientes, se puede crear un Fuzzy Extractor a partir de un Secure Sketch con la ayuda de lo que es llamado un strong extractor, cuya definición es la siguiente:

Strong Extractor: Sea $SD\ (A,B) = \frac{1}{2}\sum_v |Pr(A=v) - Pr(B=v)|$ la *Distancia estadística entre dos distribuciones.* Se define $Ext: \{0,1\}^n \rightarrow \{0,1\}^l$ usando *r* bits de aleatoriedad como un strong extractor si para todas las distribuciones $W \in \{0,1\}$ que poseen una entropía mínima *m* se cumple que $SD\left((Ext(W;X),X),(U_l,X)\right) \leq \epsilon$ donde *X* es la distribución uniforme.

Para este trabajo, un strong extractor que cumpla la condición previamente expresada son las funciones de Hash Universal, las cuales pueden entregar hasta $l = m - 2\log\left(\frac{1}{\epsilon}\right) + O(1)$ bytes de información. Esto debido a que en esta implementación, el largo de la salida de un strong extractor no es de excesiva importancia, por lo que no es necesaria una implementación más compleja que intente minimizarla.

Entonces, un fuzzy extractor creado a partir de un secure sketch y un strong extractor es:

1. Generación de P y R a partir de W: $Gen(w;r,x): P = (SS(w;r),x), R = Ext(w;x)$
2. Re-generación de R a partir de P y W' cercano a W: $Rep\left(w',(s,x)\right): w = Rec(w',s) R = Ext(w;x)$

Donde *x* es la aleatoriedad del strong extractor y *s* es el valor que se obtiene del secure sketch. En la ilustración 3 a continuación se muestra la composición de las dos funciones a partir de los componentes explicados.

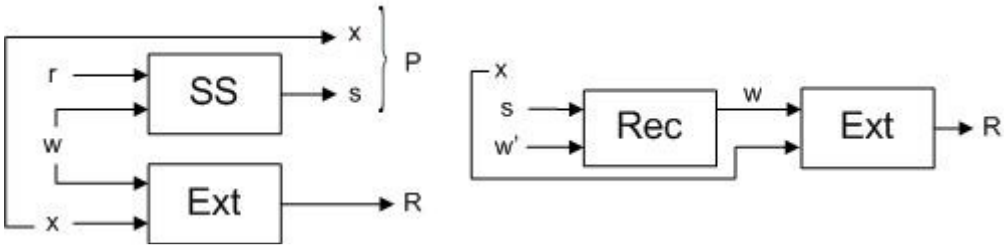

Ilustración 3: Funciones de un Fuzzy Extractos. Izq: Generar. Der: Reproducir

Como se mencionó antes, para este trabajo se usará una implementación de secure sketch llamada "pinsketch" que trabaja sobre conjuntos de tamaño variable de la siguiente forma:

Para todo elemento $w \subseteq GF(2^m)^*$ sea x_w su vector de características, donde cada posición es 1 si el elemento está en el conjunto y 0 si no está. Entonces para calcular SS(w):

1. Sea $s_t = \sum_{x \in w} x^t$ todo esto sobre $GF(2^m)$.
2. De como dato de salida $SS(w) = (s_1, s_3, s_5, \dots, s_{2t-1})$

Mientas que para calcular Rec(w',s)

1. Calcular s' = SS(w')
2. $\sigma_t = s'_t - s_t$

3. Calcular $supp(v)$ tal que $s(v) = (\sigma_1, \sigma_3, \dots, \sigma_{2t-1}) \wedge |supp(v)| \leq t$ donde $supp(v)$ es la lista de las posiciones de v con elementos distintos de 0.

4. Si $dis(w, w') \leq t \Rightarrow supp(v) = w \Delta w'$. Luego $w = w' \Delta\, supp(v)$.

Lo que hace la función Rec es calcular la diferencia entre ambas entradas w a partir de los vectores s que crea la función SS y con ese dato busca el conjunto que corresponda a la diferencia simétrica de w. Luego simplemente invierte la operación y eso permite recuperar totalmente el conjunto w

3.3. Procesamiento de huellas

Las huellas digitales se componen básicamente de dos elementos:

Líneas: En una imagen corresponden a las zonas negras o grises de la huella, son las partes en relieve y las que se usan para la obtención de las minucias y singularidades. Son relativamente paralelas en todo su recorrido a través de la huella.

Valles: Las zonas claras de la imagen, corresponden a las contrapartes de las líneas en bajo relieve.

A partir de las marcas que dejan estos dos elementos al obtener la "foto" a través del lector de huellas, se obtiene la lista de minucias después de un proceso que consta de varias fases [12]:

Obtención del mapa de orientaciones: La orientación de las líneas de una huella digital corresponde al ángulo entre el borde interno de la línea y el eje horizontal. La obtención de estos datos se hace recorriendo la imagen pixel por pixel, y luego en un segundo recorrido en bloques de n x n, calculando el ángulo dentro del bloque. La SDK usada ocupa bloques de 8 x 8. La ilustración siguiente muestra una huella digital con su mapa de orientaciones superpuesto donde se puede ver que, salvo en la zona central de la huella, donde se encuentran las singularidades, todas las orientaciones se mantienen a lo largo de la línea.

Ilustración 4: Mapa de orientaciones de una huella digital

Detección de Singularidades: las singularidades son zonas de la huella donde la orientación de las líneas cambia. Son clasificadas en dos tipos: núcleos y deltas. La combinación de las singularidades y su posición es un criterio de clasificación de huellas que sirve de primer filtro, además de patrón de identificación cuando la imagen obtenida es de mala calidad, usualmente debido a ser una huella latente o huella "muerta" (huella levantada desde un objeto).

La obtención de singularidades se hace a través de llamado operador de Poincaré que corresponde a la suma de las diferencias entre el punto a calcular y cada uno de sus vecinos, es decir, si el operador logra un valor de 360° o 180° entonces el punto corresponde a un núcleo, si logra un valor de -180° es un delta y cualquier otro valor corresponde a punto no singular.

Binarización: Con los mapas y las singularidades obtenidas se puede procesar la huella para la obtención de las minucias. El primer paso para esto es extremar los colores de la imagen, de forma de dejar las líneas en negro y los valles en blanco. Esto con el fin de facilitar el reconocimiento de las minucias más adelante. La ilustración muestra el resultado de la misma huella mostrada anteriormente después del proceso de binarización.

Ilustración 5: Binarización de una huella digital

Adelgazamiento: Se reducen las zonas negras de la imagen hasta obtener líneas de un pixel de grosor. Esto con el fin de poder encontrar minucias que de otra forma estarían ocultas por las zonas más gruesas, como es el caso de las islas o los poros. Esto se logra a través de filtros sucesivos aplicados a la imagen binarizada hasta lograr el ancho deseado. Para esto es importante mantener la conexión entre todos los puntos de una línea, sin embargo esto puede llevar a la aparición de falsas minucias, usualmente bifurcaciones e islas, por lo que es necesario tener cuidado. La ilustración 6 muestra la huella adelgazada y lista para la detección de las minucias.

Ilustración 6: Adelgazamiento de la huella digital.

Detección de minucias: Esta fase consiste en recorrer la imagen en bloques de 3 x 3 pixeles usando distintas máscaras para detectar la presencia de minucias y eliminando los falsos positivos. De

las minucias se obtienen varios datos, siendo los más importantes para este trabajo la posición y el ángulo de la minucia. Otros datos que se almacenan son la calidad y el tipo de minucia. La información lograda se almacena en un arreglo de bytes según lo indicado en la norma ISO 19.974-2. La composición de este arreglo consta de 4 partes: Los primeros 24 bytes corresponden a información del lector de huellas usado. Los siguientes 5 bytes corresponden a información de la lectura obtenida, entre ellos el dedo registrado, la calidad de la lectura y la cantidad de minucias detectada. A continuación viene una sección de 6 bytes por minucia obtenida, donde se almacena la posición, el ángulo y la calidad de la imagen. El registro finaliza con una sección opcional que lleva información extra, como la cantidad de minucias según tipo y otros datos que son usados por las SDK existentes.

Capítulo 4: Metodología

Para cumplir los objetivos planteados, en una primera etapa se recopiló el material necesario para realizar los estudios y evaluaciones correspondientes, así como para obtener un conocimiento adecuado del tema. Posterior a esto, se realizaron diversas evaluaciones enfocadas en el aspecto teórico a las propuestas encontradas, determinando cuáles de ellas son las más convenientes para el problema propuesto.

Una vez realizados los estudios, se procedió a implementar las partes del proyecto por separado, con el fin de ver su funcionamiento en la práctica, sus limitaciones y requisitos. Obteniendo resultados satisfactorios se continuó con el desarrollo de una librería integrada, manteniendo las partes independientes para permitir su exportación a otros proyectos.

Luego de verificar que la librería completa funciona adecuadamente, se continuó con la adaptación para los dispositivos BUG, partiendo con la instalación del ambiente en los dispositivos y ajustándolos a las necesidades del problema y siguiendo con el desarrollo de una aplicación para los dispositivos que use el trabajo realizado, en la forma de una comunicación bidireccional entre el dispositivo y un computador.

El mayor desafío a tener en cuenta en esta parte son las limitaciones de los dispositivos BUG, que usan un ambiente de desarrollo bastante limitado (Java PhoneME) el cual debió ser reemplazado por OpenJDK para permitir el desarrollo del proyecto.

4.1. Evaluación de las propuestas

La primera etapa en el desarrollo del proyecto consistió en la evaluación de las diversas alternativas para armar el proyecto. Debido a que se sabía que el ambiente final es limitado, el trabajo se concentró en buscar una solución ligera y rápida, buscando alternativas tanto en trabajos nuevos como más antiguos.

Finalmente, se decidió por el trabajo de Boneh y Franklin [2] para el esquema de encriptación, Paterson [7] para autentificación y Dodis, Reyzin y Smith [5] para la criptografía fuzzy. Además de eso, se complementó esta última parte con implementaciones de Hash Universal y Códigos MAC para verificar la correctitud de los resultados antes de entregar la información privada almacenada en el dispositivo. En el tema de las huellas digitales, se decidió finalmente por la SDK de Griaule Biometrics[12] debido a que es intuitiva de usar y no requiere una conexión permanente a internet. Al principio se había considerado usar la librería librfprint[13] que es de código abierto, sin embargo se desistió de ese camino al ver que para los lectores nuevos no había soporte y la librería no mostraba avances desde el año 2004.

En el diseño se consideró el hecho de que los usuarios no estarían en contacto permanente con la entidad central, por lo que se considera el funcionamiento en tres fases: La primera cuando se levanta el sistema se ejecuta una sola vez y ahí se generan los parámetros maestros del sistema de encriptación y firma. La segunda fase, cuando un usuario nuevo ingresa al sistema, consiste en el enrolamiento de su huella digital, la generación de su clave por el servidor central, la encriptación de dicha clave y el almacenamiento de los datos en el dispositivo BUG. Finalmente, para cada vez que el usuario ingrese al sistema a partir de entonces se considera la consulta de la huella digital, comparación contra el patrón almacenado y si corresponde la autentificación de los mensajes que sea emitidos. Esto logra que las

consultas a la entidad central sean mínimas y aun así permitir una comunicación autentificada entre los miembros.

Durante todo el desarrollo del proyecto, se debió tener en cuenta a qué tipo de dispositivos iba dirigido. Se debió mantener el código lo más ligero posible sin que afecte a la seguridad ni la eficiencia del trabajo además de considerar los desafíos propios de trabajar con códigos ya hechos. Varias partes necesarias del proyecto ya habían sido desarrolladas, por lo que se trabajó bastante unificando interfaces entre lenguajes (Java y C++ a través de JNI) y entre módulos.

4.2. Implementación de la solución

Para poder llegar a la implementación final se debió considerar tres fases del proyecto: Implementar cada módulo por separado, integrarlos en una aplicación en un ambiente de desarrollo normal y finalmente adaptar la aplicación para que funcione en el ambiente de los BUG.

4.2.1. Implementación de los componentes

Fuzzy Extractor: Para esta componente se trabajó con la implementación pinsketch antes descrita [5] y dos funciones principales: generar la información aleatoria, y recuperar los datos con información nueva. Ambas partes usan las dos funciones que se entregan en la implementación: Sketch que calcula el sketch de un conjunto de datos dado y Differ que, a través de un conjunto y el sketch de otro calcula la diferencia simétrica entre ambos conjuntos. La primera funcionalidad se compone de los siguientes pasos:

1. Obtener el secure sketch de los datos entregados, a través de la función SS del algoritmo secure sketch.

2. Obtener el hash de los datos obtenidos por el secure sketch. Este hash corresponde a la clave de encriptación para la información privada y se obtiene a través de una implementación de hash universal.

3. Obtener el código MAC de la clave, para su comparación con futuras consultas. Con la existencia de este código, es innecesario el almacenamiento de la clave y este dato puede ser público.

4. Encriptación de la clave privada de firma d_{id} a través de una implementación AES usando como clave el hash obtenido.

5. Almacenamiento en el BUG de los siguientes datos: código MAC, sketch de la huella patrón, texto cifrado de la clave, semilla del hash universal. El sketch de la huella se almacena en un archivo, mientras que todos los otros datos se almacenan en otro archivo aparte.

6. Eliminación del registro de la huella patrón.

Esta función se ejecuta una sola vez por usuario, al momento del registro en el sistema y es ejecutada por el servidor central. La segunda funcionalidad, el recuperar los datos y la clave privada, ese ejecutada por el usuario en su dispositivo cada vez que vuelve a ingresar al sistema y consta de los siguientes pasos:

1. Obtener la diferencia simétrica entre los datos nuevos y el sketch de la huella patrón. Esto es posible gracias a las propiedades de la función Rec del secure sketch.

2. Obtención del sketch del conjunto obtenido a través de la función SS.

3. Se vuelve a aplicar la función Rec, esta vez entre los datos nuevos y el sketch de la diferencia simétrica. Esto devolvería lo que debería ser el conjunto de datos original.

4. Se ejecuta por última vez la función sketch para obtener la información del último conjunto obtenido, y se eliminan los datos nuevos.

5. Se obtiene el hash de la información lograda del paso anterior, usando como semilla del hash universal la almacenada en el dispositivo.

6. Se obtiene el código MAC del hash, y se compara con el almacenado en el momento del registro, si estos coinciden, entonces la clave derivada es correcta y la identidad corresponde al usuario permitido por el dispositivo.

7. Si la clave es la correcta, se desencripta el texto cifrado y se le entrega al usuario, para luego eliminar los datos del hash obtenidos.

Las Ilustraciones 7 y 8 muestran el diagrama de flujo de las dos funcionalidades.

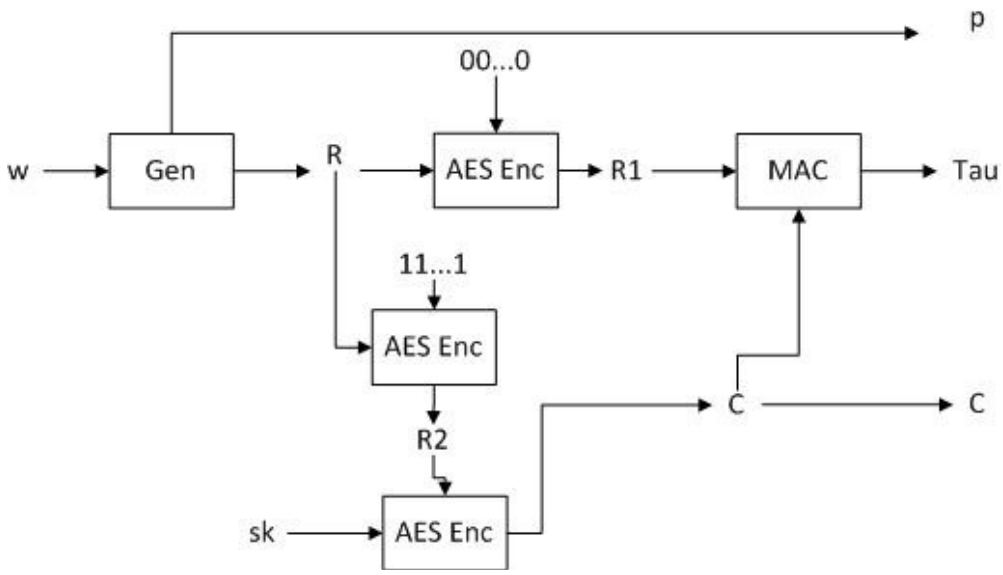

Ilustración 7 Esquema de la generación de parámetros del Fuzzy Extractor

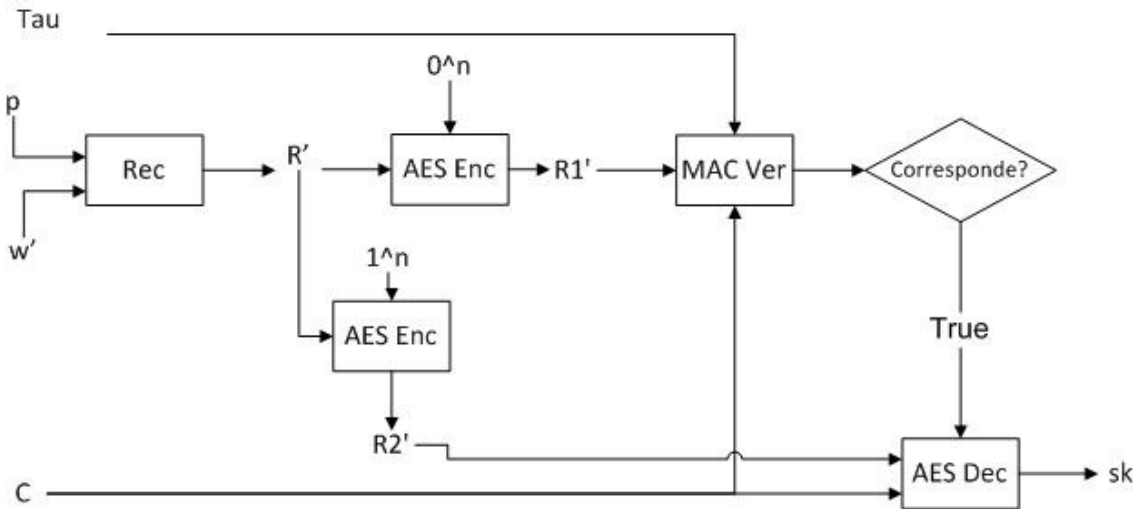

Ilustración 8: Esquema de la recuperación de parámetros del Fuzzy Extractor

Autentificación basada en Identidad: Este módulo trabaja en dos partes. La primera la maneja el servidor central con las funciones de inicialización y generación de claves mientras que la segunda es manejada por los usuarios y contiene las funciones de firma y verificación.

Las funciones constan de los siguientes pasos

Inicialización:

1. Se carga la curva elíptica con la que se va a trabajar
2. A partir de la curva se obtienen los siguientes parámetros
   a. $P \in \mathbb{G}$ generador del grupo.
   b. $s \in \mathbb{Z}_q^*$ aleatorio
   c. Funciones de hash: $H_1: \{0,1\}^* \rightarrow \mathbb{G}^*$, $H_2: \{0,1\}^* \rightarrow \mathbb{Z}_q$, $H_3: \mathbb{G} \rightarrow \mathbb{Z}_q$
   d. Se calcula la clave pública: $P_{pub} = sP$

Generación de claves:

1. Se calcula la clave pública $Q_{id} = H_1(id)$ y privada $d_{id} = sQ_{id}$ del usuario a través de su identidad *id*.
2. Se entrega al usuario su clave privada mas los parámetros públicos del esquema $mpk = \langle P, P_{pub}, H_1, H_2 \rangle$

Las funciones de Firma y Verificación no tienen cambios con respecto a lo explicado en la parte 3.1.1 de este trabajo.

Procesamiento de huellas digitales: Esta componente funciona igual tanto en el servidor central como en los dispositivos BUG y funciona a través de los datos brutos que son entregados por la SDK usada.

1. Desde los datos, se extraen las minucias según lo descrito en el estándar ISO [14] y se almacenan en una estructura adecuada.

2. Se calcula la diferencia entre cada par de minucias y se almacena. Se consideró la diferencia entre minucias como la diferencia de cada componente por separado: X, Y y ángulo. La razón de esto es evitar falsos negativos en el reconocimiento debido a mala postura del dedo, rotaciones o incluso que alguna minucia haya desaparecido de la imagen, debido a una cicatriz o a que el lector está sucio por ejemplo.

3. Se respalda la información obtenida para su uso por otros módulos.

4.2.2. Integración de las componentes.

Para poder integrar todas las partes, y aún así lograr que cada módulo fuera independiente de los otros, se decidió el mantener cada código externo lo más parecido a su versión original posible, manteniendo sus interfaces. Debido a esto, la comunicación entre el módulo de huellas y el fuzzy extractor es a través de archivos que almacenan los datos mientras que la comunicación entre éste y el módulo de encriptación es simplemente a través de variables.

Se escogió esta alternativa a pesar de la potencial brecha de seguridad que existe con la presencia del archivo, basado en el hecho de que ese proceso se ejecuta esporádicamente y el archivo es eliminado apenas es posible, minimizando la ventana de vulnerabilidad.

4.2.3. Adaptación a ambiente BUG

Esta fase incluyó el desarrollo de una aplicación exclusiva para los BUG, ya que estas tienen una estructura propia que debe ser respetada. Debido a las limitaciones del ambiente, hubo que hacer algunos cambios principalmente con respecto a las estructuras de datos usadas en la versión general de la solución. Además, en el dispositivo se instaló exclusivamente las rutinas que le corresponden, separando efectivamente el proyecto en dos partes: el servidor central y el cliente.

Además se incluyó en esta fase del desarrollo dos aplicaciones, una que simula el servidor central del sistema encargado de la generación de las claves y el registro de usuarios, y por otro lado otra que sea un servidor el cual recibe los datos de los BUG para su proceso.

Capítulo 5: Descripción del Sistema de Autentificación

5.1. Arquitectura

La implementación final de la solución se puede ver en la siguiente figura, donde se muestran las tres componentes principales del sistema con los módulos que cada una necesita

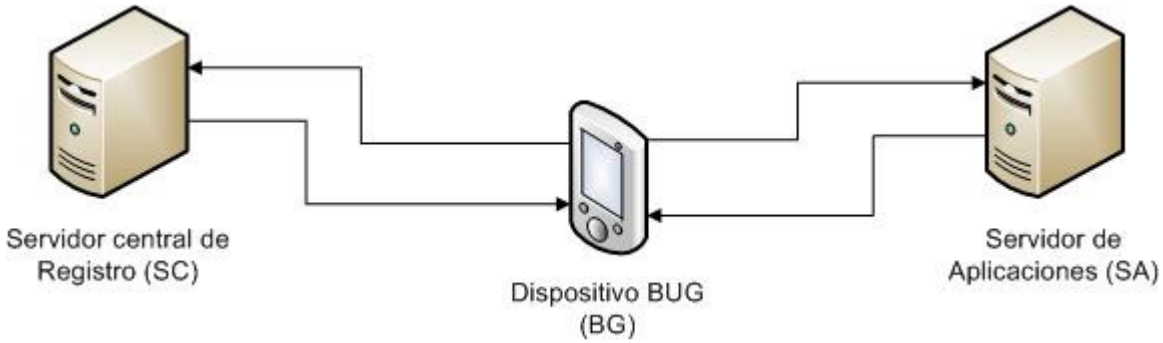

Ilustración 9: Mapa General del sistema de Autentificación

Servidor Central (SC): Esta componente es la encargada de inicializar los parámetros del sistema de autentificación, además de generar las claves para los usuarios y extrae la información de sus huellas digitales al momento del registro, así como almacenar las identidades de los dispositivos que está siendo usados y a quien están asignados. Para este ejemplo se considera un solo servidor aunque perfectamente puede ser un grupo de servidores para redes con mucho tráfico de usuarios, aunque dado que el trabajo de esta componente es esporádico no se considera necesario.

BUG (BG): En este esquema, como ejemplo se consideró un solo dispositivo BUG, aunque en la realidad serán varios los BUG que interactúan entre sí. Esta componente es la que obtiene información del ambiente, trabajo que es simulado en esta implementación. Esa información es autentificada y enviada ya sea a la red ad-hoc o al servidor que la está esperando a través de internet. Además aquí se debe identificar el usuario cada vez que ingrese al sistema después de su registro, para poder hacer uso de su información privada.

Servidor de aplicación (SA): Esta parte, simulada por una aplicación en la implementación, corresponde al servidor que estará fijo en alguna parte del sistema, esperando la información recolectada por los dispositivos BUG para su procesamiento. Aquí, al igual que en los BUG se corren las rutinas de verificación y firmas de mensajes para la comunicación de los dispositivos, además de almacenar una lista con los dispositivos autorizados.

5.2. Descripción General

El proceso completo, desde la inicialización del SC hasta la recepción y verificación de un mensaje por parte del SA, a grandes rasgos consiste en

Inicialización: La entidad central, usando SC genera los parámetros maestros del sistema, tanto los de autentificación como los de extracción de información de las huellas

Registro: Cuando un usuario nuevo es registrado, se ingresa su identidad y la del dispositivo que le será asignado, así como la huella digital del usuario. Con esos datos SC extrae la información necesaria de la huella digital a través del fuzzy extractor y luego de generar la clave privada de autentificación la encripta con la clave respectiva. El texto cifrado, más los parámetros públicos de autentificación y el código MAC de la clave son guardados en el dispositivo BUG para su posterior uso.

Ingreso: Una vez registrado el usuario, debe identificarse cada vez que ingresa al sistema a través de su huella digital. La aplicación BG verifica la información de la huella y si corresponde libera la clave privada del usuario para la firma de mensajes.

Firma: Cada mensaje que sea enviado a la red de BUGs o a la aplicación SA es firmado con la clave obtenida en la fase anterior, usando el algoritmo descrito en la parte 3.1 de este documento. Luego de esto el mensaje es liberado a la red.

Verificación: Cuando un mensaje llega a SA, lo primero que hace es verificar que la identidad del remitente sea una de las identidades autorizadas para la red. Si es así, se procede a la verificación de la firma y el mensaje a través del algoritmo mostrado en la parte 3.1 de este trabajo.

5.3. Detalles de la implementación

La implementación, en todas sus fases se hizo con una mezcla de lenguajes C++ y Java, comunicándose entre sí a través de la interfaz JNI. Esto debido a que la implementación de pinsketch estaba realizada en C++, mientras que el lenguaje de desarrollo por defecto para los BUG es Java, además de ser el lenguaje de implementación de la SDK de huellas digitales disponible para Linux.

Además de eso, se usaron varias librerías disponibles en internet para distintas partes del proyecto:

- JPBC: Librería que maneja todas las operaciones basadas en pares bilineales para su uso en criptografía.

- NTL: Librería en C usada para cálculos de alta precisión numérica, usada por pinsketch

- Gnu.crypto: Librería de código abierto java que maneja varias primitivas criptográficas, cifradores de bloque y hash universal entre ellas

- simplerestclient: Implementado especialmente para dispositivos BUG, esta librería es una implementación sencilla de un cliente para webservice REST, con todas sus operaciones básicas.

Por otro lado, se crearon clases para la representación de los pares bilineales y las minucias, además de una clase que representa los parámetros públicos del sistema, llamada PKG.

El proyecto completo se dividió en dos librerías: una para SC y otra para cualquier miembro del sistema, ya sea BUG o SA. Cada librería contiene todo lo necesario para su correcto funcionamiento, incluyendo manejo de huellas digitales, fuzzy extractors y el esquema de firmas, debido a que se puede hacer un corte limpio entre las funcionalidades necesarias para SC y las para los miembros. La ilustración 10 es un mapa conceptual de la división de funcionalidades por paquetes para cada una de las entidades. El nodo raíz del árbol se usa simplemente como nombre y no implica que ambas librerías, representadas

en los dos sub árboles principales estén conectadas de alguna forma. Los nombres en itálico corresponden a las clases implementadas.

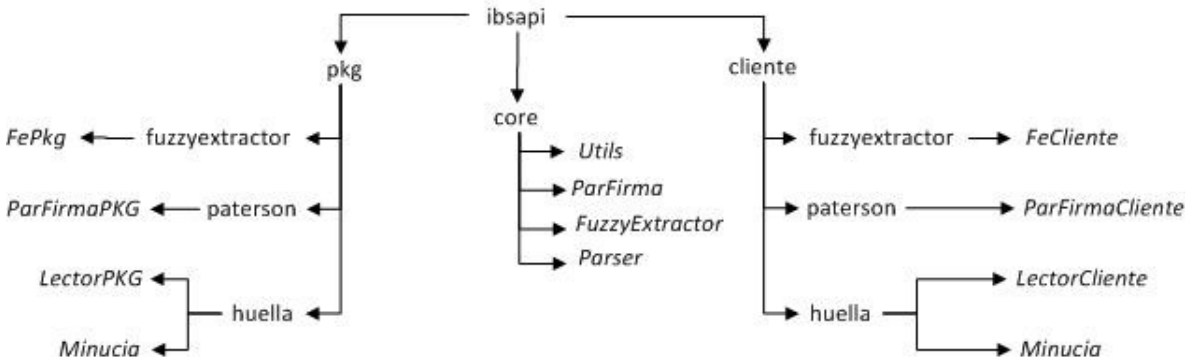

Ilustración 10: Esquema conceptual de clases del proyecto

Esta división de funcionalidades tiene como consecuencia cierto grado de redundancia, ya que hay partes que son necesarias para ambas librerías (contenidas en el paquete *core*, perteneciente a ambas librerías). Sin embargo esto es preferible a permitir que, debido a tener todas las funcionalidades en una sola librería, el SC pueda firmar a nombre de algún miembro o alguien pueda generar claves legales con los parámetros públicos del sistema. A continuación se da una lista de las clases implementadas, con una breve descripción de lo que hacen:

- core.Utils: Contiene funciones de soporte al sistema, como salvar datos a un archivo, cargar datos de un archivo, etc.

- core.ParFirma: Clase Singleton que implementa las funcionalidades comunes del esquema IBS para el cliente y el servidor SC, principalmente la inicialización de la curva elíptica y las funciones de hash usadas.

- core.FuzzyExtractor: Clase Singleton que implementa la creación del FuzzyExtractor, común tanto para SC como para los clientes del sistema.

- core.Parser: Clase que implementa un parser XML, usado en la comunicación de mensajes entre los BUG y entre ellos y SA. Esto debido a que los BUG funcionan a través de web services tipo REST

- cliente.fuzzyextractor.FeCliente: Clase Singleton que implementa el Fuzzy Extractor específico para lo BUGs y para SA. Hereda de la clase FuzzyExtractor del paquete core e incluye la función Recuperar

- cliente.huella.LectorCliente: Clase que implementa todo lo necesario para el manejo de las huellas digitales tanto en los BUG como en SA.

- cliente.huella.Minucia: Clase usada para representar las minucias obtenidas de la huella digital.

- cliente.paterson.ParFirmaCliente: Clase Singleton que hereda de ParFirma, donde se incluye los algoritmos de firma de mensajes y verificación de firmas.

- pkg.fuzzyextractor.FePkg: Clase Singleton que implementa el Fuzzy Extractor específico para SC, heredando de la clase FuzzyExtractor del paquete core. Incluye la función Generar

- pkg.huella.LectorHuella: Clase donde se implementan todo lo necesario para el manejo de huellas digitales al momento del enrolamiento del usuario.

- pkg.huella.Minucia: Clase usada para representar las minucias, al igual que la clase del mismo nombre del paquete cliente.huella.

- pkg.paterson.ParFirmaPkg: Clase Singleton que hereda de ParFirma, donde se implementan los algoritmos de Setup y Generación de claves del esquema de firmas.

5.3.1. Módulos Principales programados

Inicialización del sistema (FePkg.java, ParFirmaPkg.java); Aquí se inicializan todos los parámetros del sistema, tanto para la generación de las claves de los usuarios, como del fuzzy extractor para la obtención de información de las huellas digitales. Los parámetros públicos son almacenados en el archivo "parametros.properties" donde después se agregará la clave de los usuarios que se vayan registrando y se transferirá al dispositivo para su uso. Las principales funciones usadas son:

- public void InitFE(); Inicializa todos los parámetros necesarios del fuzzy extractor: instancia AES, UMAC, UHASH.

- public void InitIBS(): Inicializa los parámetros maestros y públicos del sistema de firmas, para la posterior generación de claves

Creación de claves (FePkg.java, ParFirmaPkg.java, LectorPkg.java); Aquí se inicializan las claves de los usuarios cuando se registran al sistema. El servidor toma su huella digital, la procesa y obtiene la clave privada de identificación a través de la función Generar del fuzzy extractor. Por otro lado crea la clave privada de autentificación con su identidad y la cifra con la primera clave obtenida. Este dato, mas el código MAC de la clave de identificación y los parámetros públicos del sistema son grabados en un archivo llamado "parametros.properties" y enviado al dispositivo BUG. Las principales funciones usadas aquí son

- public void FormatearData(byte[] raw); Extrae las minucias desde los datos brutos entregados por el lector de huellas en formato de un arreglo de bytes, calcula las diferencias entre minucias y las guarda en el archivo que será usado por la función generar

- public void Salvar(String path); Salva todos los datos públicos del usuario en el archivo "config.properties" y lo envía a path en el dispositivo BUG. luego elimina el archivo.

- public void Generar(String path); Genera un sketch de la huella digital a partir de los datos del archivo almacenado en path.

Firma de mensajes (ParFirmaCliente.java); Este módulo, implementado dentro del paquete cliente.paterson, es el que contiene las rutinas necesarias para el proceso de firma de un mensaje. Los parámetros públicos son cargados desde el archivo que fue entregado al momento del registro, mientras

que la clave privada es obtenida desde el mismo archivo y el módulo de autentificación de usuarios. Este módulo se apoya en las librerías de XML de java para formatear de manera sencilla los datos, de forma que puedan ser enviados a por la red de una forma estándar. Sus principales funciones son:

- public String Firmar(String msg); Firma el mensaje entregado por parámetro según el algoritmo explicado en la sección 3.1 de este trabajo y lo ingresa en un formato XML, junto con la identidad del remitente para su posterior verificación.

- public boolean Verificar(String msg); A partir del string entregado como parámetro que viene en formato XML, se obtienen el mensaje, la firma y la identidad del remitente y se verifican según el algoritmo mostrado en la sección 3.1 de este documento. Si la firma corresponde al mensaje y a la identidad la función devuelve el valor "true", siendo "false" la respuesta en caso contrario.

Autentificación de usuarios (LectorCliente.java): Este módulo es el que se encarga de que la persona que está usando el BUG (o que esté a cargo de SA) sea quien dice ser, para invalidar el sistema en caso contrario, tanto en el servidor de aplicaciones como en el dispositivo BUG. Trabaja con la SDK de Griaule para el lector de huellas digitales y con la implementación del fuzzy extractor. Lo que sucede aquí es que el usuario pone su dedo, a partir de las minucias de su huella es generado un archivo el cual pasa al módulo Recuperar del fuzzy extractor (el cual fue cargado al iniciar la aplicación a partir del archivo enviado en el momento del registro), el cual si corresponde devuelve la clave privada del usuario. Las principales funciones son:

- public void FormatearData(byte[] raw); al igual que en el caso de generación de claves, este método es el que toma los datos en bruto y los procesa para obtener las minucias, sus diferencias y grabarlas temporalmente en un archivo.

- Public String Recuperar(String path); Tomando el archivo entregado en path, esta función ejecuta el algoritmo mostrado en la sección 3.2 luego del cual se recuperaría el conjunto patrón (el usado en el momento del registro) si la huella es la correcta. Con ese resultado, se obtiene la clave simétrica usada para proteger la clave privada del sistema de firmas y si es correcta se desencripta la información privada y se eliminan la clave simétrica y el archivo temporal, para finalizar devolviendo la clave al usuario.

Capítulo 6: Usos y desafíos

Un sistema como el desarrollado, en ambiente de prototipo y general tiene aplicaciones en tantas áreas como sea necesario, por ejemplo:

- El método propuesto en este trabajo para el procesamiento de huellas puede ser un gran aporte. Partiendo por el hecho de que, si almacenar una huella digital hoy en día abarca aproximadamente 700 bytes únicamente en datos necesarios de la huella, esta propuesta da la alternativa de almacenar toda la información en no mas de 20 bytes cuando mucho. El proceso es ligero de repetir, por lo que no existe un sobre costo con respecto a los sistemas actuales en tiempo de procesamiento y, teóricamente más seguro al usar exclusivamente información pública, recreando la información confidencial cuando sea necesario, por lo que no existe en el sistema información que se pueda usar para que alguien pueda falsificar una huella sin la huella misma.

- El sistema completo puede usarse en muchos ámbitos, desde monitoreo de acceso permanente, haciendo listas con las distintas identidades que están autorizadas para cada parte de una planta (por ejemplo) combinándolo con él módulo GPS que es posible incluir en los BUG se tiene un monitoreo constante y a grandes rasgos a tiempo real de donde está cada usuario del sistema.

- Se puede transmitir información sensible de forma autentificada y confidencial (si se implementa el esquema de encriptación estudiado) por ejemplo en hospitales, donde se podría saber exactamente que doctor o que enfermera tomó los datos de qué paciente, dejando la existencia de una trazabilidad de mensajes a la hora de hacer consultas o auditorías, incluso si la información es recolectada de forma automática a través de redes de sensores, con estos últimos en las camas de los pacientes registrando lo necesario para su tratamiento.

Entre los desafíos, quizás el más imperante es medir la eficiencia del sistema y de ser posible mejorarla. Este trabajo tenía como objetivo ver si el sistema podía ser implementado de forma funcional en un ambiente restringido como son los BUG, por lo que aún queda toda la optimización por hacer.

Otro desafío importante es hacer el estudio en un ambiente mas realista, probablemente una red pequeña de prototipos con un Gateway, con el fin de ver el comportamiento del sistema con mas de una identidad existente. No debería haber grandes diferencias, pero es algo que se debe hacer.

Capitulo 7: Conclusiones

Con respecto a la lista de objetivos enunciada al inicio de desde documento se puede concluir:

- Efectivamente es posible implementar un esquema de autentificación ligero en datos y en procesamiento, que además permita un funcionamiento en localidades donde no se puede acceder a la entidad central y que permita el uso de entidades externas o el acceso de las mismas a datos del sistema. Sin embargo el proceso es enteramente interno y mucho mas fácil de comunicar que el tradicional. Si se quiere hacer la analogía con el esquema tradicional basado en certificados, el rol de éste último lo vienen a cumplir por un lado la clave privada que se le entrega al usuario al momento del registro y por otro lado los parámetros públicos del sistema que son entregados a cada miembro y en los cuales confían estos últimos el correcto funcionamiento del sistema.

- El estudio de las herramientas usadas permitió lograr una combinación funcional de las mismas, la cual hasta ahora no se había visto. Ninguna de las herramientas usadas es de las mas nuevas en su área, pero los trabajos independientes mostraron ser en la fase de estudio los mas ligeros de implementar.

- Los esquemas basados en identidad, tanto en encriptación como en autentificación, son variados y cada uno con sus ventajas y desventajas, por lo que es posible cambiar de esquema, dependiendo de las circunstancias de uso, para obtener un mejor rendimiento o si se piensa, una mayor seguridad.

- El almacenamiento estandarizado de la información que nuestras huellas digitales contienen, permite un uso transversal de hardware y del software especializado, sin entrar en desmedro de la calidad de información almacenada ni de su posterior uso. Esto permite el desarrollo independiente de las herramientas a usar, lo que da más robustez al sistema.

- Aparte de los objetivos logrados, el implementar un sistema como éste requiere de un uso a conciencia de los dispositivos. Prototipos o no, el usar biometría de forma constante en cualquier ambiente requiere de una actitud responsable de los usuarios, no por el hecho de que intenten encontrar una forma de evitar las medidas, sino porque es necesario entender que en la mayoría de los casos, esas mismas medidas serán para la propia seguridad de quienes están en el sistema, y a veces eso es difícil de comprender, más aun si implica trabajo extra.

- Aunque se ha avanzado mucho desde los inicios de la criptografía basada en identidad, se puede ver que aun hay mucho campo tanto para investigación como para implementaciones de prototipo. La prueba de esto es la cantidad de variantes que se encontraron y que se siguen proponiendo, cada una con características diferentes y aptas para distintos escenarios. Claramente esta área se encuentra en pleno auge.

Referencias

[1] BAEK, J., NEWMARCH, J., SAFAVI-NAINI, R., SUSILO, W. 2004. "A survey of Identity-Based Cryptography" Proceding of AUUG'04. Melbourne, Australia. Organization for Linux, Unix and Open Source Professionals. pp 10p.

[2] BONEH, D., FRANKLIN M. 2001. "Identity-based Encryption from the Weil pairing". En Advances in Cryptology: Proceedings of CRYPTO '01. Santa Bárbara, USA. International Associaton for Cryptograhpic Research. Springer. pp 31p

[3] CHATTERJEE, S., SAKAR, P. 2009. "Identity Based Encription and Hierarchical Identity Based Encription" En: JOYE, M., NEVEN, G. Identity-Based Cryptography, Vol 2. Países Bajos, pp 45-64.

[4] COCKS, C., 2001. "An Identity Based Encryption Scheme Based on Quadratic Residues". En Cryptography and Coding, 8th IMA International Conference. Cirencester, Reino Unido. Springer. pp 3p.

[5] DODIS, Y., REYZIN, L., SMITH, A. 2004. "Fuzzy Extractors: How to generate strong keys from biometrics and other noisy data" Advances in Cryptology: Proceedings of EUROCRYPT' 04. Interlaken, Suiza. Springer. pp 47p.

[6] KATZ, J., LINDELL, Y. 2007. "Digital Signature Schemes" En: Introduction to Modern Cryptography. Primera Edicion. USA. Chapman & Hall/ CBC Press. pp 421 – 456.

[7] PATERSON, K. 2002. "Identity-Based Signatures from Pairings on Elliptic Curves" Electronic Letters Vol 38(18): 1025-1026.

[8] SAHAI, A., WATERS, B. 2005. "Fuzzy Identity-Based Encryption" Proceedings of EUROCRYPT' 05. Aarhus, Dinamarca. International Association for Cryptographic Research. Springer. pp 15p.

[9] SHAMIR, A. 1984: "Identity-Based Cryptosystems and Signature Schemes". En: Advances in Cryptology: Proceedings of CRYPTO '84. Santa Bárbara, USA. International Association for Cryptographic Research. pp 7 p.

[10] BUGLabs. http://www.buglabs.com [consulta: 10 septiembre 2010]

[11] GRIAULE BIOMETRICS. Griaule Biometrics homepage: http://www.griaulebiometrics.com [consulta: 25 agosto 2010]

[12] GRIAULE BIOMETRICS. Book: Understanding Biometrics: http://www.griaulebiometrics.com/page/en-us/book/understanding-biometrics [consulta 03 septiembre 2010]

[13] INTERNATIONAL ORGANIZATION FOR STANDARIZATION. ISO/IEC 19794-2:2005/Amd 1:2010 - Detailed description of finger minutiae location, direction, and type: http://www.iso.org/iso/catalogue_detail.htm?csnumber=52537 [consulta 15 octubre 2010]

[14] LIBFPRINT: Libfprint project homepage: http://reactivated.net/fprint/wiki/Libfprint [consulta: 03 julio 2010]