Universidad Autónoma del Estado de México

Centro Universitario UAEM Zumpango

Generación de información única del usuario basada en biometría dinámica de tecleo implementado en Android

TESIS

que para obtener el título de

INGENIERO EN COMPUTACIÓN

presenta:

Jesús Israel Reynoso Coronel

Dr. Asdrúbal López Chau

Asesor

Zumpango, Estado de México Mayo, 2014

Dedicatoria

Gracias a mis padres por brindarme
la oportunidad de estar aquí,
de terminar mis estudios, como
muchos otros quisieran hacerlo.

Papá, gracias por trabajar tan duro
y tanto tiempo como lo has
hecho para que podamos seguir
estudiando.

Mamá, gracias por estar siempre para nosotros,
por ser la confidente de mis locuras y bueno,
por muchas cosas más.

Gracias al apoyo de mi familia me es posible estar en esta etapa
de mi vida, a su motivación y apoyo incondicional
en mis decisiones erróneas y acertadas es que
pude terminar este trabajo.

Gracias a mis profesores por darme la oportunidad de
participar en el proyecto, por el que obtuve una beca de apoyo
a la investigación, de la misma manera por brindarme el

espacio para trabajar con este tema.

Gracias al proyecto UAEM 3424/CHT2013 por la beca otorgada durante el periodo junio 2013 a mayo 2014.

De la misma manera agradezco a mi asesor Dr. Asdrúbal López Chau y mis revisores M. en T.I. Carlos Alberto Rojas Hernández y Dr. Arturo Redondo Galván por el tiempo dedicado a este trabajo.

Resumen

Actualmente, los sistemas informáticos requieren cada vez mayor seguridad en el control de acceso a los usuarios y a la prestación de servicios a los que están autorizados. Para lo cual, los esquemas criptográficos son ampliamente utilizados a través de los algoritmos de autenticación y cifrado. Por otro lado, las técnicas de biometría dinámica pueden ser utilizadas para identificar individuos, y ofrecer mayor control en el acceso.

En este trabajo, se presenta un método que usa la dinámica de tecleo del usuario para producir información única por cada acceso y al mismo tiempo, el sistema no permite asociar al usuario con esa información única generada por él.

El sistema propuesto se denomina SIGENIU (Sistema de Generación de Información Única) implementado para dispositivos móviles con sistema operativo Android. El sistema obtiene el tiempo individual del usuario cada vez que pulsa una tecla en la pantalla, al resolver un captcha. La información obtenida es procesada en un vector de características único que finalmente se presenta en una gráfica sobre el dispositivo móvil de tal manera que el usuario observe su comportamiento.

La unicidad de la información producida con la biometría informática puede ser utilizada en los esquemas criptográficos para ofrecer la atenticación ya que estos esquemas requieren variables únicas y aleatorias, por ejemplo, las firmas digitales que utilizan números aleatorios denominados llaves de sesión.

Es importante destacar que la finalidad de la técnica es generar información única a partir del comportamiento del usuario. Esta información debe de ser lo suficientemente diferente para cada uno, de tal manera que lo distinga de los demás usuarios, pero que no necesariamente lo identifique, ya que lo esencial en la técnica propuesta es producir información única. Esto es diferente a lo propuesto en otros trabajos o sistemas del estado del arte.

Abstract

Currently, the computer systems require increasingly security access control to users and the provision of services to which they are authorized. For which purpose cryptographic schemes are widely used through authentication and encryption algorithms. On the other hand, dynamic biometric techniques can be used to identify individuals, and provide greater control access.

In this work, a method that uses the user keystroke dynamics to produce unique information at each access is presented. At the same time, the system must avoid to associate the behavior of the users' identity. In order to display results, the Android operating system was used.

The proposed system is called SINEGIU (Sistema de Generación de Información Única), which captures the time for each character typed to solve the captcha.The information captured is processed in a unique vector of features which is presented in a graph over a mobile device where the user can see his or her behavior.

The information produced with dynamic biometric can be used in a cryptography schemes to offer authentication, due to, this kind of schemes require variables that must be unique and random, as example, the digital signatures that use random numbers called the session key.

It is important to note that the main goal of our technique is to generate unique information from the behavior of the user. This information must be different for each user such that the user can be different to other users, however, the technique can not identify to the user. This is different to other proposals presented in the literature.

Contenido

Lista de Tablas

Lista de Figuras

Capítulo 1

Introducción

En la actualidad el uso de los sistemas informáticos es cada vez mayor debido al fácil acceso a la información a través de ellos. Tareas comunes como el control del personal en una empresa, la vigilancia en negocios, la contabilidad, etc., son automatizadas en sistemas que se utilizan únicamente por personas autorizadas, en el mejor de los casos. Es común que se utilice un mecanismo de identificación, como un nombre de sesión y una contraseña, para acceder a estos sistemas.

Uno de los problemas que ocasiona este tipo de identificación, es el descuido por parte de los usuarios, quienes muestran u ofrecen su información confidencial (usuario y la contraseña) sin saber que mantenerla en secreto es toda la seguridad con la que cuenta el sistema. Así, cualquier persona que conozca esa información puede acceder al sistema, usurpando la identidad de un usuario.

Afortunadamente, existen mecanismos robustos para la identificación de personas como la autenticación y la biometría informática que se pueden definir, de la siguiente manera:

La *autenticación* es un servicio de seguridad relacionado con la identificación de entidades o de datos, es decir, una entidad tiene que comprobar que es quien dice ser o se debe corroborar la validez de los datos, lo que implica la integridad de los mismos [24].

La biometría informática es un conjunto de técnicas utilizadas para la identificación de personas basadas en uno o más aspectos físicos, así como en aspectos conductuales, es

decir, se miden características de las personas de manera directa o indirecta, aplicando técnicas de clasificación, selección, agrupamiento o detección de casos anómalos, etc. [18].

En la mayoría de los casos, la autenticación y la biometría han sido aplicadas de forma independiente. Ejemplo de ello son los sistemas de voto electrónico, donde se han propuesto protocolos seguros de votación utilizando firmas digitales o sistemas biométricos como la huella digital para la identificación de los votantes.

No obstante, si ambas técnicas fueran utilizadas tanto para autenticar como para generar información única de los usuarios, es posible mejorar la seguridad de los sistemas, con una probabilidad insignificante de vulnerabilidad en ataques de usurpación de la identidad.

1.1. Planteamiento y definición del problema

En algunos esquemas criptográficos de llave pública, como las firmas digitales, es necesaria la generación de números aleatorios para proteger la información. La herramienta más utilizada para realizar tal tarea es la función generadora de números pseudoaleatorios.

Los equipos de cómputo cuentan con una función generadora de números pseudoaleatorios que usa un número semilla. Uno de los grandes inconvenientes para la función generadora es que la semilla suele ser la misma para los equipos con igual modelo y marca, lo que ocasiona que se generen números repetidos en equipos físicamente diferentes.

Lo anterior es un problema mayor para los esquemas criptográficos ya que una parte de la seguridad de dichos algoritmos es la utilización de números aleatorios, pero para cumplir con dicho requisito es necesario utilizar números semilla diferentes para cada llamada a la función generadora, esto dará como resultado un número pseudoaleatorio único.

El principal problema radica, entonces, en la generación de números semilla únicos para ser utilizados durante la llamada a la función generadora.

1.2. Hipótesis

Como se mencionó anteriormente, es un requisito que los números pseudoaleatorios sean generados de forma única, para ello se necesita que los números semilla sean diferentes. Considerando que el ser humano no realiza dos actividades exactamente de la misma manera y tomando ventaja de la biometría dinámica, es posible generar información única a partir del comportamiento del usuario. En este contexto, el término información única, se refiere a que la probabilidad de repetición es despeciable. Tal información puede ser utilizada de diferentes formas tales como un arreglo de diferentes tipos, un número grande, un argumento para una función picadillo o como semilla para funciones generadoras de números pseudoaleatorios, esto último implicaría generar números diferentes en cada llamada a dichas funciones.

1.3. Justificación

La unión de la biometría dinámica y la criptografía de llave pública para la generación de números pseudoaleatorios basados en el comportamiento del usuario, supone una mejora significativa en la seguridad de los sistemas de información.

La generación de números pseudoaleatorios diferentes para cada usuario, no significa la creación de un nuevo algoritmo. Es más bien, tener un mecanismo que permita generar una semilla con un valor diferente para cada número generado de forma pseudoaleatoria por el usuario.

Los números semillas, al ser generados a partir del comportamiento del usuario aplicando técnicas de biometría dinámica, permitirán la obtención de números pseudoaleatorios diferentes y únicos para cada usuario. Lo que implica que los sistemas sean robustos en la defensa de ataques como usurpación de la identidad.

Además, un mecanismo de generación de información única, que llamamos de manera general *semilla* puede ser utilizado para diferentes aplicaciones como por ejemplo en un sistema capaz de identificar a un usuario a través de su interacción con el, o en sistemas donde se requiere información única como por ejemplo folios o etiquetados, o sencillamente para analizar el comportamiento del usuario en diferentes estados de

ánimo.

1.4. Objetivos general y específicos

1.4.1. Objetivo General

Implementar un mecanismo basado en biometría dinámica para obtener un número semilla y poder generar números pseudoaleatorios únicos.

1.4.2. Objetivos Específicos

- Investigar sobre las técnicas de biometría dinámica.
- Proponer un mecanismo que use la biometría dinámica para obtener información única del usuario.
- Implementar el mecanismo propuesto con biometría dinámica en diferentes infraestructuras tales como PC y móviles para generar números semilla.
- Utilizar los números semilla en funciones de generación de números pseudoaletorios.
- Analizar la unicidad de los números pseudoaleatorios generados.
- Implementar un algoritmo de firma digital utilizando los números semillas generados.

1.5. Alcance del proyecto

En este proyecto se tiene como meta generar semillas únicas basadas en biometría dinámica de tecleo sobre dispositivos móviles con sistema operativo Android en sus versiones 2.2 en adelante. Las semillas obtenidas son utilizadas como argumentos para las funciones generadoras de números pseudoaleatorios. Su funcionalidad es comprobada con un análisis de unidad de las semillas generadas.

1.6. Organización del documento

El resto del documento se describe a continuación. En el capítulo 2, se presentan los fundamentos de la biometría dinámica, la criptografía de llave pública, así como de los números pseudoaleatorios. En el capítulo 3 se presenta el diseño del sistema propuesto. En el capítulo 4 se detalla la implementación de sistema, mientras que en el capítulo 5 se presentan los resultados y las pruebas realizadas. Finalmente, en el capítulo 6 se listan las conclusiones y el trabajo futuro.

Capítulo 2

Fundamento teórico

En este capítulo se presentan los conceptos básicos de la criptografía, la biometría y los dispositivos móviles, para tener un mejor entendimiento de los temas expuestos en el resto del documento.

2.1. Preliminares

En la actualidad, con la infraestructura de cómputo al alcance de todos, los ataques informáticos son cada vez más exitosos. Un *ataque* es definido como una amenaza llevada a cabo de forma exitosa que permite una violación de la seguridad. Un *adversario* es la entidad, humano o computadora, que realiza el ataque. Existen ataques pasivos y activos. Los primeros son aquellos donde sus acciones no alteran o afectan al sistema, mientras que los segundos afectan directamente con la funcionalidad del sistema [1].

Algunos de los ataques más conocidos son los siguientes:

- Intercepción: es un ataque pasivo utilizado para observar la información transmitida entre las redes de comunicaciones.
- Interferencia: es un ataque pasivo donde el adversario detecta cierta información analizando patrones en el tráfico de la red.
- Intrusión: es un ataque activo donde el adversario consigue acceder a los recursos o a la información privada del sistema.

- Engaño: es un ataque activo que atenta contra la información del sistema, se divide en:
  - Usurpación: el adversario utiliza la información privada de algún usuario válido para tener libre acceso al sistema a nombre de él.
  - Falsificación: el adversario altera o fabrica información que el sistema considera como válida.
  - Repudio: el adversario puede negar que recibió o envió cierta información.

Para evitar esos ataques, a lo largo de la historia, se ha buscado mantener la privacidad en la comunicación entre dos o más personas. Proteger la información ante un tercero es el objetivo de la criptografía, donde se han propuesto muchos esquemas de seguridad para distintas aplicaciones.

El uso de técnicas criptográficas tiene como propósito prevenir las fallas de seguridad en un sistema electrónico. La seguridad en general, se refiere al cumplimiento de los servicios de seguridad que se listan a continuación [24]:

- *Confidencialidad* garantiza que la información privada pueda ser accedida únicamente por las entidades autorizadas.
- *Integridad* dedicada a la protección de los datos para que no sean modificados, destruidos o extraviados de una manera maliciosa o accidental.
- *Autenticación* la autenticación es una comprobación de que la entidad es quien dice ser.
- *No rechazo* asegura que el remitente de cierta información no pueda rechazar/negar su transmisión o contenido y que el receptor no pueda negar su recepción o contenido.

De acuerdo a la finalidad del sistema a proteger, los cuatro puntos anteriores tanto en la teoría como en la práctica deben ser evaluados considerando lo siguiente [36]:

- El *nivel de seguridad* que es una cota inferior en términos de las operaciones necesarias para alcanzar un objetivo, usando los mejores métodos conocidos. En

otras palabras, es el número mínimo de operaciones requeridas para romper la seguridad de una primitiva dada.

- La *funcionalidad* indica la necesidad de combinar varias técnicas criptográficas para cumplir todos o la mayoría de los servicios de seguridad en un cripto-sistema.

- Los *métodos de operación* son las diferentes formas en las que las primitivas pueden ser implementadas, dependiendo de su funcionalidad.

- El *rendimiento* se refiere a la eficiencia de la primitiva, en términos de tiempo de ejecución, tamaño del código en el caso de bibliotecas de software y área del circuito en caso de implementaciones en hardware, para un modo de operación en particular.

- La *facilidad en la implementación* requiere el mínimo de dificultad para poner en práctica una primitiva, ya sea en un ambiente de software o de hardware.

La criptografía se divide en dos categorías: de llave secreta y de llave pública. El objetivo fundamental en los dos tipos es mantener la comunicación segura entre dos entidades en un canal que es inseguro como la Internet.

Los esquemas de llave secreta se caracterizan por el hecho de usar una llave secreta para ocultar y descubrir la información que se desea proteger. Los esquemas de llave pública utilizan una llave pública y una llave privada, donde la pública se deriva de la privada y ésta última está protegida matemáticamente por problemas computacionalmente difíciles de resolver.

Dado que, en este trabajo se tiene especial interés en el servicio de seguridad de autenticación, se describe más detalle a continuación.

2.2. Autenticación

Con el creciente avance tecnológico en los dispositivos móviles y el incremento en el uso de éstos en la vida diaria, se ha despertado un gran interés en el desarrollo

de aplicaciones móviles que deben ser óptimas en procesamiento y almacenamiento, además del acceso a información a través de un canal inalámbrico.

El hecho de que las aplicaciones tengan muchas ventajas como el acceso a los datos, entretenimiento, lectura de diversos archivos, etc. tienen el problema de ser inseguros por el medio por el cual se comunican, que es el aire. Para evitar que los datos sean vulnerables a intrusos, se tiene la necesidad de utilizar mecanismos de seguridad que protejan los sistemas que requieren de un intercambio de información y un acceso seguro a la información privada. Dicho lo anterior, la criptografía es una herramienta que ayuda a disminuir las posibles amenazas utilizando la *autenticación* como servicio de seguridad más relevante. la cual se puede describir de formas más detallada de la siguiente manera [15]:

Autenticación: permite certificar que la identidad de las entidades participantes en la comunicación es verdadera. Se logra verificando dichas entidades usando mecanismos como firmas digitales, certificados digitales o características biométricas, de acuerdo a los siguientes tres factores:

1. **Algo que el usuario tiene**: puede ser un dispositivo difícil de clonar, al mismo tiempo tiene que ser común y sencillo para los usuarios. Por ejemplo un teléfono inteligente ya que tiene poder de cómputo y así es posible almacenar información protegida contra manipulaciones.

2. **Algo que el usuario sabe**: como una contraseña lo suficientemente compleja, mezclando mayúsculas, minúsculas y dígitos, además no debe tener ninguna relación con el usuario respecto a sus asuntos personales, profesionales o familiares.

3. **Algo que el usuario es**: como las huellas dactilares, el tono de voz, la forma de escribir o patrones oculares.

El último factor está relacionado directamente con la biometría informática mientras que los dos primeros están relacionados con la criptografía de llave pública. En las siguientes secciones se presentan los mecanismos de autenticación basados en biometría informática y en firmas digitales.

2.3. Biometría

El concepto de biometría (bio:vida, metría:medida) se refiere al conjunto de métodos y equipos (generalmente electrónicos) utilizados para medir o identificar alguna característica física o conductual de las personas, para el reconocimiento único.

La biometría informática aplica métodos basados en técnicas matemáticas para la autenticación o verificación de la identidad de los usuarios de sistemas, con la finalidad de mejorar las tecnologías de seguridad. Se puede clasificar en estática y dinámica. La primera consiste en utilizar mediciones de la anatomía del sujeto, mientras que la segunda usa medidas del comportamiento. Las mediciones más comunes en la biometría informática estática son: la huella digital, iris, retina, rasgos faciales y venas en algunas partes del cuerpo. Para el caso de la biometría informática dinámica, algunos ejemplos de mediciones son: la forma de escritura, voz, movimientos corporales y patrón de escritura en un teclado de computadora [13, 18, 27].

2.3.1. Biometría Estática

Es la parte de la biometría que se encarga del estudio de las características biológicas que son singulares e inalterables a lo largo del tiempo, además de ser intransferibles a otros individuos. Algunas aplicaciones de la biometría estática son:

1. Reconocimiento de la persona por medio de las huellas dactilares. Esto se realiza mediante un lector de huellas, la información es procesada y almacenada, posteriormente se comparan algunos de los patrones que se obtuvieron con información guardada en algún medio de almacenamiento. La Figura 2.1 muestra un ejemplo de lo descrito anteriormente.

Figura 2.1: Digitalización de una huella digital a través de un lector [32].

2. El reconocimiento del individuo por rasgos faciales. El procedimiento para lograr el reconocimiento del individuo es similar al del reconocimiento por medio de las huellas dactilares (véase la Figura 2.2).

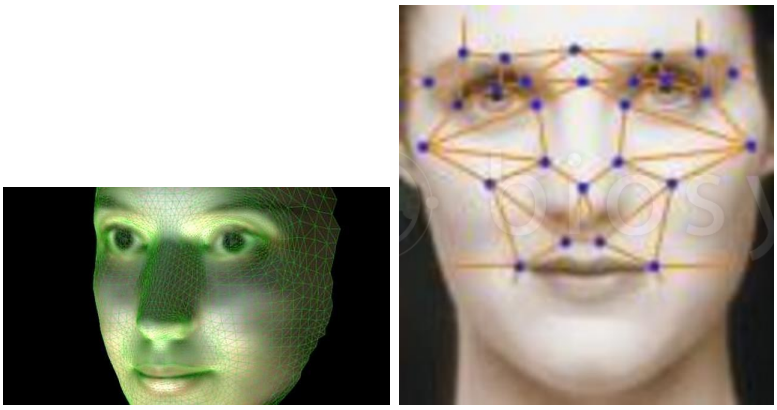


Figura 2.2: Resultado de la digitalización de un rostro humano [33].

3. Reconocimiento de manos. El procedimiento consiste en comparar las medidas de las manos considerando largo y ancho de los dedos, dimensiones de la palma de la mano, la medida de las diferentes articulaciones, etc., con este análisis se logra identificar las manos del usuario. Éste método es muy aceptado ya que es muy económico al necesitar únicamente de dos fotografías ortogonales de la mano de frente y de los lados de la mano. El proceso es realmente sencillo, sólo se hacen mediciones de lo alto y ancho de la mano además de la medida de los dedos. A pesar de ser muy sencillo para la identificación de los usuarios, en lugares donde se requiere tener un control de acceso para personal no es muy utilizado, ya que tiene algunas fallas dependiendo de las actividades que el individuo realice.

4. Reconocimiento del iris. Este sistema se basa en patrones retinales, los cuales

son obtenidos mediante una cámara que saca una foto en blanco y negro. Para que la información sea precisa y concisa se debe contar con una la iluminación adecuada para que el análisis se pueda realizar de manera correcta, cuando el sistema obtiene la foto del ojo se realiza una serie de deformaciones retinales y a su vez deformaciones matemáticas las cuales brindarán información necesaria para autenticar o no al usuario dentro del sistema, (Figura 2.3).

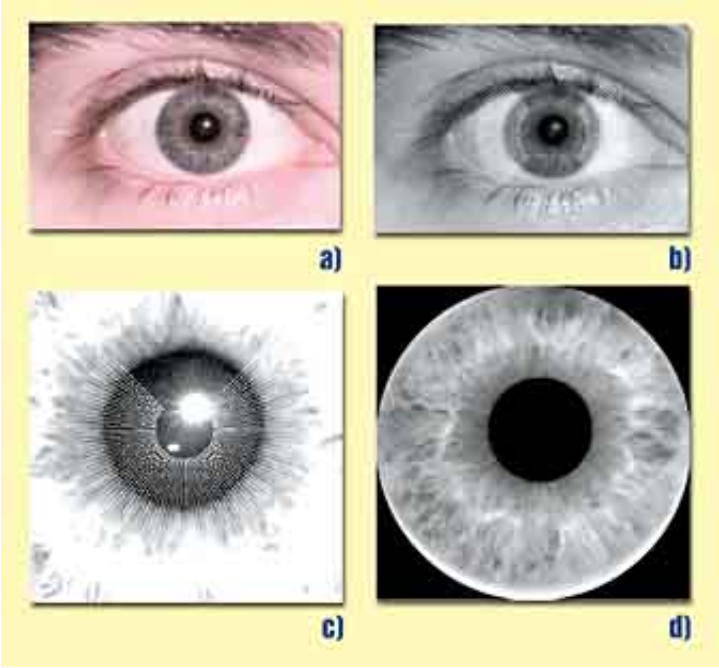


Figura 2.3: Resultado de la digitalización de un ojo humano y la identificación de patrones [34].

2.3.2. Biometría Dinámica

Recientemente han llamado la atención a la comunidad científica las técnicas de biometría dinámica que no requieren equipo adicional al que ya posee la mayoría de las personas, como es el caso de las computadoras, teléfonos celulares y las tabletas [5]. En estos dispositivos, se han desarrollado métodos para generar información a partir de las características de los usuarios, por ejemplo, a través de interfaces de usuario gráficas [31, 10], gestos generados por el usuario tanto en pantalla o en el aire [12, 14, 19] , así como dinámica de tecleo. Esta última resulta atractiva ya que todos los dispositivos

modernos incorporan uno, ya sea físico o virtual.

Ya que la biometría es el estudio de las mediciones del cuerpo humano (tiempo de reacción a diversas situaciones, características biológicas que son particulares e inalterables, etc.), la información obtenida después de aplicar algunos estudios a los individuos es procesada, el resultado de este proceso se puede utilizar para diferentes áreas, una de ellas es la criptografía. Aplicar los datos de la biometría dinámica a la criptografía (algoritmos de cifrado de datos) garantiza que el usuario al que se le realizó el estudio biométrico es el dueño de los datos que serán únicos para cada participante debido a su singular comportamiento [27].

A continuación se muestran de forma general algunas técnicas propuestas en la literatura para brindar autenticación a través de la biometría dinámica.

- **Reconocimiento de firma**

  Uno de los objetos de estudio de la biometría dinámica es la firma manuscrita ya que es una característica de comportamiento que refleja el estado de ánimo de un individuo. La verificación de la firma es un método usado desde la antigüedad, a pesar de ello, se ha cuestionado la veracidad del método por las posibles falsificaciones, por lo cual se requiere de diversos factores para poder comprobar su autenticidad como son: tiempo en el que realiza la firma, la presión ejercida, inclinación de las líneas, el tamaño de los ciclos, las puntas, etc., conocidos todos esos factores, es demasiado compleja la reproducción exacta de las firmas.

- **Sonido de la voz**

  La voz es otra de las características que se pueden utilizar para realizar la identificación del usuario, se dice que ésta técnica puede ser empleada de diversas formas, incluso utilizando un mismo texto para las identificaciones siguientes. La técnica, que tiene variantes como el carácter de la persona, la edad, etc., consiste en analizar las señales que genera el habla, puede hacerse mediante algún dispositivo dedicado al análisis de tonos de la voz.

- **Dinámica de tecleo**

Esta técnica ha sido estudiada desde hace muchos años. En la segunda guerra mundial, la inteligencia militar de Estados Unidos analizaba el "ritmo"de los mensajes enviados por código Morse [6]. En [20] existe un resumen sobre sistemas y métodos que utilizan técnicas basadas en la dinámica del tecleo. En [16] se analiza el efecto de las características medidas de la dinámica de tecleo sobre la precisión de clasificación. El sistema presentado en [23] utiliza la fase de registro en sistemas informáticos para construir un clasificador y detectar intrusos. En [2] se desarrollan técnicas para evaluar las dinámicas de tecleo continuas, que son utilizadas para monitorear al usuario durante su interacción con el sistema, y no únicamente durante el acceso. Con la finalidad de brindar mayor protección a los sistemas que utilizan biometría, en [26] se propone utilizar gráficos como contraseñas, incorporando una medida extra en la presión del dedo del usuario sobre pantallas táctiles.

2.4. Firmas digitales

La firma digital tiene la misma esencia que el de una firma autógrafa, con el servicio adicional de proteger la información de modificaciones ya sea intencionales o accidentales. En la criptografía moderna, los esquemas de llave pública son ampliamente usados para generar firmas digitales. La llave privada del usuario, la cual debe ser conocida sólo por su propietario, es requerida por el firmante para producir una única e infalsificable firma digital para un documento dado, mientras que la llave pública debe ser conocida por el verificador de la firma, con la finalidad de decidir si la firma del documento es válida o no.

Considerando la siguiente notación:

- $\diamond$ $\mathcal{M}$ representa el conjunto de todos los mensajes que pueden ser firmados
- $\diamond$ $\mathcal{S}$ representa el conjunto de todas las firmas que pueden ser generadas, usualmente con una longitud fija
- $\diamond$ $\mathcal{K}_{\mathcal{S}}$ representa el conjunto de llaves privadas

- $\diamond$ $\mathcal{K}_{\mathcal{V}}$ representa el conjunto de llave públicas
- $\diamond$ $\mathcal{S}_{\mathcal{E}} : \mathcal{M} \times \mathcal{K}_{\mathcal{S}} \to \mathcal{S}$ representa las reglas de transformación para que la entidad $\mathcal{E}$ produzca una firma
- $\diamond$ $\mathcal{V}_{\mathcal{E}} : \mathcal{M} \times \mathcal{K}_{\mathcal{V}} \to \{verdadero, falso\}$ representa las reglas de verificación de la transformación para una firma producida por $\mathcal{E}$. Estas reglas son usadas por las entidades que requieran la verificación de la firma

Una firma digital se puede definir de la siguiente forma [24]:

Un esquema de firma digital es una tripleta de algoritmos (*Genera, Firma, Verifica*) tales que,

i. *Genera* es el algoritmo de generación de llaves. Tiene como entrada un parámetro de seguridad ℓ y como salida un par $(k_{\mathcal{S}}, k_{\mathcal{V}}) \in \mathcal{K}_{\mathcal{S}} \times \mathcal{K}_{\mathcal{V}}$, correspondiente a la llave privada y pública, respectivamente.

ii. *Firma* es el algoritmo de firma. Tiene como entrada $(m, k_{\mathcal{S}}) \in \mathcal{M} \times \mathcal{K}_{\mathcal{S}}$ y como salida un elemento $\sigma \in \mathcal{S}$, el cual es la firma del mensaje m producida con la llave privada $k_{\mathcal{S}}$.

iii. *Verifica* es el algoritmo de verificación. Tiene como entrada $(m, \sigma, k_{\mathcal{V}}) \in \mathcal{M} \times \mathcal{S} \times \mathcal{K}_{\mathcal{V}}$ y como salida un valor *verdadero* para indicar una firma válida o un valor *falso* en caso contrario. Se dice que una firma es válida si para cualquier tripleta $(m, \sigma, k_{\mathcal{V}})$ se cumple que

$$Verifica(m, Firma(m, k_{\mathcal{S}}), k_{\mathcal{V}}) = verdadero$$

para cada $(k_{\mathcal{S}}, k_{\mathcal{V}})$ obtenido del algoritmo *Genera* y para cada $m \in \mathcal{M}$.

En las firmas digitales, las funciones picadillo son usadas para producir digestos de mensajes de cualquier longitud. Una función picadillo es una función de sólo ida [17] y se define de la siguiente manera [25]: *Una* función picadillo *(*hash*)* $H : \{0,1\}^* \to \{0,1\}^n$ *es una función de sólo ida que convierte una cadena binaria de longitud arbitraria a*

una cadena binaria de longitud fija. Se dice que $H(x)$ *es el digesto de* $x \in \{0,1\}^*$*, correspondiente a* H*, de longitud* n*.*

H se considera segura si los siguientes problemas son computacionalmente difíciles de resolver [25]:

La *preimagen*, es decir, dado $y \in \{0,1\}^n$, encontrar $x \in \{0,1\}^*$ tal que $H(x) = y$; la *segunda preimagen* que dado un elemento $x \in \{0,1\}^*$ se requiere encontrar un $x' \in \{0,1\}^*$ tal que $x \neq x'$ y $H(x) = H(x')$ y las *colisiones*, esto es, encontrar $x, x' \in \{0,1\}^*$ tal que $H(x) = H(x')$.

Además debe satisfacer dos propiedades muy importantes: ser eficiente y garantizar que un cambio, de incluso 1 bit, a la entrada produzca una salida diferente.

Para efectos de eficiencia, una firma digital es producida para el digesto del mensaje. Las Figuras 2.4 y 2.5 muestran el funcionamiento de un esquema de firma digital. El remitente usa su llave privada k_S para producir la firma σ. En la verificación, únicamente si la firma fue producida con la llave privada correspondiente a la llave pública con la que se está verificando y el mensaje es el originalmente firmado, el resultado de la verificación será aceptable.

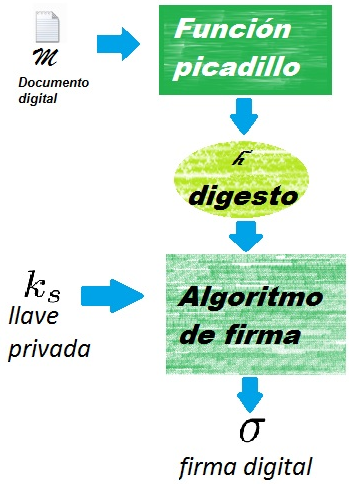


Figura 2.4: Procedimiento de Firma.

Figura 2.5: Procedimiento de Verificación.

La seguridad de los esquemas de firma digital depende de la construcción del par de llaves y de la producción de firmas genuinas. En ambos casos es necesario el uso de

números aleatorios, los cuales, en todas las ocasiones son generados de forma pseudo-aleatoria a través de funciones generadoras. Por lo anterior es necesario prestar atención a los números aleatorios y pseudoaleatorios.

2.4.1. Números Aleatorios y Pseudoaleatorios

Los números aleatorios son generados al azar, es decir, que todos los números dentro de un rango tienen la misma posibilidad de ser elegidos. Éste tipo de números sólo pueden ser generados por los humanos ya que simplemente se escoge un valor sin la necesidad de utilizar alguna operación matemática para seleccionarlo [28].

Los números pseudoaleatorios son generados por un algoritmo en algún dispositivo electrónico, los métodos realizados requieren de un número semilla al que se le realizaran diversas operaciones matemáticas para generar un número pseudoaleatorio diferente en cada iteración.

Dos de los algoritmos que ya existen para la generación de números pseudoaleatorios son los siguientes [9]:

Método del Cuadrado Medio: Éste método necesita de un numero semilla el cual es elevado al cuadrado y posteriormente se extraen los dígitos del centro, dicho número es el generado por la función.

Método de Congruencia Lineal: Éste método genera una secuencia de números enteros desde 0 hasta $m-1$ de la siguiente manera:

$$X_{i+1} = (a * X_i + c) mod\, m,\ \ i = 0, 1, 2, ...$$

donde X_0 es llamado semilla, a es el multiplicador y m el módulo.

Como puede observarse, la utilización de los números semilla son altamente importantes para que la generación de números pseudoaleatorios no produzca dos valores iguales.

Debido a que muchas de las aplicaciones informáticas requieren la utilización de números pseudoaleatorios, los fabricantes de hardware considera una función pseudo-aleatoria en cada equipo electrónico, inclusive en los dispositivos móviles. Por tal razón

y además del interés de utilizar los dispositivos móviles en este trabajo, en la siguiente sección se presenta una breve descripción del sistema operativo móvil más utilizado en la actualidad: Android [35].

2.5. Sistemas operativos para móviles

Los teléfonos celulares, los smartphones y algunos otros dispositivos cuentan con un SO, el cual les permite ejecutar las aplicaciones o simplemente interactuar con el dispositivo para realizar alguna tarea. Existen varios SO con versiones en cada uno, por ejemplo está el Symbian que fue un SO producto de la alianza de varias empresas de telefonía móvil, entre las que se encontraban Nokia, Sony Mobile Communications, Psion, Samsung, Siemens, Arima, Benq, Fujitsu, Lenovo, LG, Motorola, Mitsubishi Electric, Panasonic, Sharp, etc. Sus orígenes provenían de su antepasado EPOC32, utilizado en PDA's y Handhelds de PSION.

Otro SO es IOS que es un sistema operativo móvil de la empresa Apple Inc. Dicho SO se desarrolló para ser utilizado en el iPhone, tiempo después lo instalaron en otros dispositivos como iPod Touch (reproductor de música), iPad (tableta electrónica), etc. La comunicación del usuario con el equipo está dado por la interfaz de usuario. La interfaz de IOS está hecha y diseñada bajo el concepto de manipulación directa, que tiene como herramienta principal los gestos táctiles y multitáctiles (movimiento en la pantalla con uno o varios dedos a la vez que se encargan de realizar diversas acciones, la utilidad de las acciones depende del contexto). Algunos elementos de control con los que cuenta el SO son los deslizadores o *scroll bar*, botones, áreas de texto, etc.

Otro SO es Windows Mobile, que desarrolló Microsoft para ser utilizado en teléfonos inteligentes (Smartphone) , al ser diseñado y desarrollado para teléfonos móviles, Microsoft se vio en la necesidad de hacer un SO compacto ya que la mayoría de los teléfonos tienen pocos recursos; Windows Mobile está basado en el kernel de Windows CE, además de contar con aplicaciones de la API de Microsoft Windows.

Bada es un SO móvil desarrollado por Samsung. Está diseñado para cubrir tanto los teléfonos inteligentes de gama alta como los de gama baja.

BlackBerry es una marca de teléfonos que desarrollo la empresa canadiense del mismo nombre, dicha empresa creo el teléfono no solo con las mismas características de un teléfono común si no que añadió el servicio de correo electrónico móvil y cumple con las funciones básicas de un Smartphone.

Android, desarrollado por Android, Inc. Está basado en Linux y fue desarrollado para dispositivos con pantalla táctil, para un pronto crecimiento este SO es de código abierto.

2.5.1. Sistema operativo Android

Existen varios tipos de sistemas operativos, por ejemplo Symbian, iOS, Windows Mobile, BlackBerry y Android.

Android es un sistema operativo de código abierto, basado en el sistema operativo Linux y desarrollado exclusivamente para dispositivos móviles con pantalla táctil. Actualmente se encuentra respaldado económicamente por la empresa Google desde el 2005 [3].

El sistema operativo está compuesto de aplicaciones principalmente desarrolladas en el lenguaje JAVA, ejecutadas en una VM de Dalvik. Las bibliotecas del lenguaje C tienen un administrador para las interfaces gráficas, un framework OpenCore, base de datos SQLite, una API OpenGL. Además, cuenta con aproximadamente 12 millones de líneas de código, de las cuales $3,000,000$ son líneas XML, $2,800,000$ son líneas de lenguaje C, $2,100,000$ de líneas Java y otras $1,750,000$ líneas en lenguaje $C++$.

Existen alrededor de diez versiones principales de Android, en este proyecto se considera desde la versión Android 2.2 Froyo (Froyo, frozen yogurt), la cual fue lanzada el 20 de mayo de 2010, lo que significa que es compatible con las siguientes características:

- Núcleo Linux 2.6.32.
- Optimizaciones en velocidad, memoria y rendimiento.
- Integración del motor de JavaScript V8 de Chrome en el navegador.

- Soporte para el servicio Android Cloud to Device Messaging (C2DM), con notificaciones cortas.
- Soporte para Microsoft Exchange, incluyendo políticas de seguridad, auto−descubrimiento.
- Funcionalidad de anclaje de red por USB y Wi−Fi hotspot.
- Opción para deshabilitar acceso de datos sobre red móvil.
- Contraseñas numéricas y alfanuméricas.
- Soporte para instalación de aplicaciones en la memoria expandible.
- Soporte para Adobe Flash.

2.5.2. Historia

El SO Android, lo desarrolló originalmente la empresa Android, Inc. y tiempo después se convirtió en el principal producto de la Open Handset Alliance (Un grupo de 84 compañías fabricantes, desarrolladoras de hardware y software entre otras actividades, que se dedican a desarrollar estándares abiertos para dispositivos móviles. Algunos de los miembros que integran la alianza son Google, HTC, Dell, Intel, Motorola, Qualcomm, Texas Instruments, Samsung, LG, T-Mobile, Nvidia y Wind River Systems [21]) ya que los teléfonos con Android son los más vendidos en los Estados Unidos, además de ser un producto a nivel mundial. Este SO se popularizó rápidamente, en 2010 tenía cubierto en $43,6$ porciento del mercado, para el 2011 ya tenía cubierto el $50,9$ porciento, más del doble que IOS, que era el segundo SO en el mercado.

Una gran población de programadores están haciendo aplicaciones, ya hay más de $1,000,000$ de éstas en Google Play, que es la tienda oficial de aplicaciones de Android. A pesar de que está basado en el núcleo de Linux no está libre de Malware, las aplicaciones pueden contener código malicioso, es por ello que Google Play tiene un analizador de código, el cual verifica que las aplicaciones estén libres de este tipo de problemas.

Android se anunció el 5 de noviembre de 2007, junto con el la creación de la Open Handset Alliance; Android, Inc. liberó una gran parte de código bajo la licencia de Apache.

2.5.3. Arquitectura de Android

Las líneas de código del SO Android, así como el cualquier otro, tiene una arquitectura la cual puede ser analizada. En la figura 2.6 se muestran las capas de la arquitectura de Android. Como se puede observar, se muestran cuatro capas, cada una depende de los servicios de la inferior.

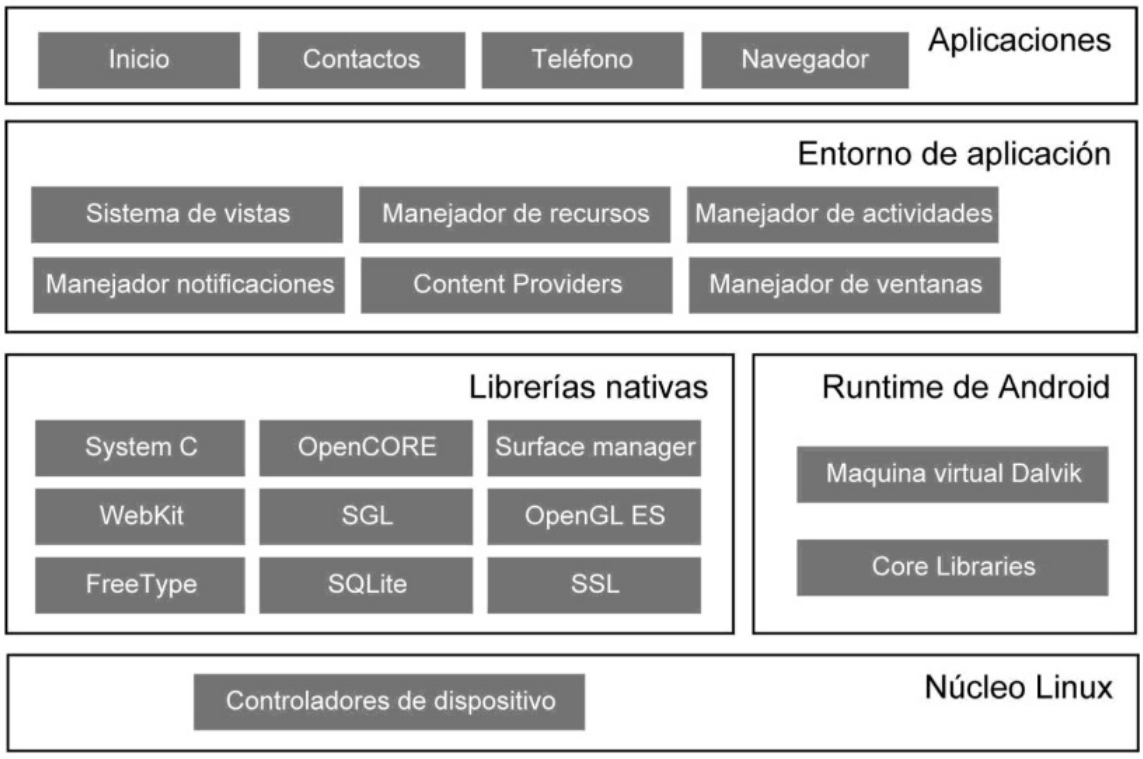


Figura 2.6: Arquitectura global del SO Android [3].

El kernel o núcleo de Linux

El sistema operativo Linux está siendo ocupado por una gran cantidad de dispositivos, pueden ser desde computadoras con poco poder para tratar la información

hasta supercomputadoras con grandes capacidades de procesamiento. El núcleo de Android es la versión 2.6 de Linux, la gestión de memoria, seguridad, el manejo de multiprocesos, etc. son servicios que proporciona propiamente esta capa.
Esta capa es la única que depende del hardware porque actúa como modelo de abstracción entre el resto de la pila y los componentes del dispositivo [4].

Tiempo de ejecución Android

Está basado en el concepto de la máquina virtual (VM) de Java, pero considerando las limitaciones de los dispositivos móviles (memoria y procesamiento) en donde se ejecuta Android no fue posible implementar una VM de Java, por ello, Google decidió crear la *VM de Dalvik* y así todo el código escrito para Android en Java se ejecutara en dicha máquina virtual. Existen dos diferencias importantes:

- La VM de Dalvik ejecuta archivos con extensión *.dex*, que son archivos generados en la compilación a partir de los archivos *.class* (*bytecode*) y los *.jar*, estos archivos están optimizados además de estar basado en registros para lograr el objetivo de Android, su implementación en un dispositivo móvil y
- Se incluye el *core libraries* con la una gran cantidad de clases y métodos de java SE

La VM de Dalvik, escrita por y diseñada por Dan Bornstein, ejecuta archivos .dex que son creados a partir del bytecode de java con instrucciones independientes, el archivo resultante es diferente del bytecode. Permite que se ejecuten varias instancias de VM al mismo tiempo y se aprovecha del núcleo para aspectos de seguridad y aislamiento de procesos [3].

Bibliotecas nativas

Contiene un conjunto de bibliotecas escritas en lenguaje C y C++, las cuales están usadas en algunos componentes de Android. Las bibliotecas del SO utilizan código bajo la licencia de OpenGL, algunas de ellas son:

System C library: Es una derivación de las bibliotecas BSD del lenguaje C, esta biblioteca fue adaptada para dispositivos embebidos basados en Linux

Media Framework: Biblioteca basada en PacketVideo's OpenCORE, soporta reproducción y grabación de videos en varios formatos

Surface Manager: Controla el acceso al subsistema de representación gráfica de 2D y 3D

WebKit: Soporta el navegador utilizado en el navegador Android y el la función *webview*

SQLite: Motor de bases de datos relacionales, similar a MySQL

SSL: Proporciona servicios de cifrado de datos (Secure Socket Layer)

Entorno de aplicación

Proporciona un ambiente de desarrollo con diversos componentes como pueden ser sensores, localizadores, barras, etc. esta capa fue diseñada con la finalidad de reutilizar los componentes, elementos que pueden o ser publicados según la seguridad del sistema lo permita. Para desarrollar aplicaciones se cuanta con la facilidad de programación del lenguaje Java. El SDK de Android no cumple todo lo que especifica el estándar pero ya cumple una gran parte de las funciones de JRE. Los servicios más destacados de la estructura son:

Views: La parte visual de los componentes

Resource Manager: Acceso a algunos recursos que no estas incluidos en el código

Activity Manager: Controla el ciclo de vida de las aplicaciones, además proporciona un sistema de navegación entre ellas

Notification Manager: Controla las notificaciones de las aplicaciones en la barra de estado

Content Providers: Mecanismo que permite el acceso a los datos de otras aplicaciones

Aplicaciones

Esta es la última capa de la arquitectura del SO Android, es en ella donde se instalan todas las aplicaciones que el programador debe de haber programado en el lenguaje java con el SDK o en el lenguaje C con NDK (Native Development Kit). El usuario interactúa con esta capa ya que es donde la VM muestra la información producida, normalmente un usuario no se da cuenta lo que ocurre en las capas inferiores, incluso no sabe de la existencia de ellas, pero un programador debe de estar consciente de su existencia, asi como de la función de las mismas.

2.5.3.1. Ciclo de vida de una aplicación Android

Una aplicación está integrada de un conjunto de elementos llamados actividades, las cuales le permitirán al usuario interactuar con ella. Las actividades son elementos que controlan el ciclo de vida, debido a que cuando un usuario utiliza la aplicación cambia de actividad, las cuales se van a almacenar en una pila de datos con el resto de las actividades para una posible navegación del usuario entre ellas, la navegación puede darse cuando el usuario presione la tecla de atrás.

Como ya se mencionó anteriormente, el núcleo del SO Android es Linux, entonces las aplicaciones se ejecutan de la misma manera que en cualquiera de las distribuciones de Linux, una aplicación en cada proceso, así la gestión de la memoria utilizada por la aplicación queda sujeta a la disposición del SO. En la Figura 2.7 se puede observar como es el ciclo de vida de una aplicación Android.

Cuando se inicia la aplicación, es el momento en el que se crea la actividad, continua la ejecución de los métodos *onCreate*(), en este método se pueden inicializar todos los elementos necesarios para la interfaz; *onStart*(), que indica que la aplicación esta lista para ser mostrada al usuario y *onResume*(), este método se llama cuando la aplicación comenzará la interacción con el usuario.

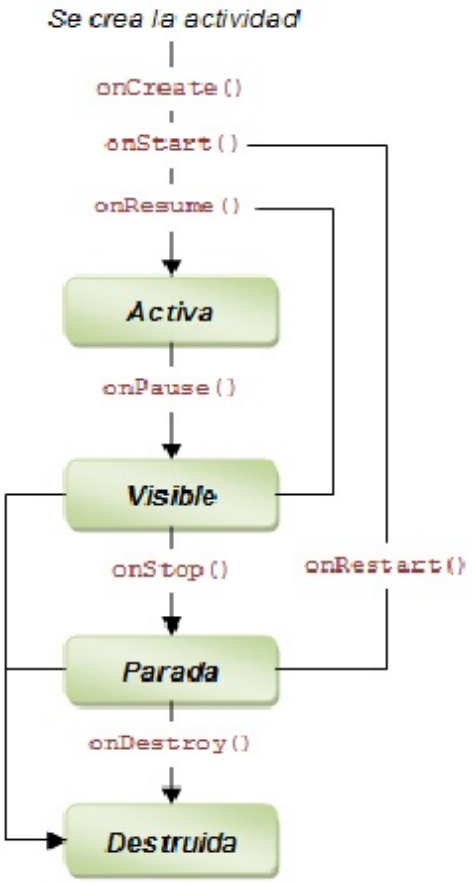


Figura 2.7: Ciclo de vida de una aplicación Andorid [29].

Activa quiere decir que la aplicación se está visualizando en pantalla y puede interactuar con el usuario. Si el usuario ejecuta otra aplicación, se llama un método llamado *onPause*() que quiere decir que la aplicación está a punto de pasar a segundo plano, es cuando la música y las animaciones propias de la aplicacion deben ser detenidas.

Visible quiere decir que la aplicación sigue en ejecución, pero hay una ventana que no ocupa toda la pantalla, como una ventana emergente que cubre parcialmente la pantalla del dispositivo, por ejemplo la ventana del mensajero del Facebook (aplicación independiente a la aplicación del Facebook).

El método *onStop*() es llamado cuando la aplicación ya no es visible para el usuario, es entonces cuando pasa al estado de *parada o detenida*.

El método *onRestart*() es llamado cuando la aplicación regresa a la pantalla, esto

quiere decir que ya había pasado por la ejecución del método *onStop*().

Parada se alcanza este estado cuando la aplicación ya no es visible para el usuario, es entonces cuando el programador debe de guardar todos los datos de la aplicación como son: El estado de la interfaz, las preferencias, datos ingresados en la aplicación, etc.

El método *onDestroy*() es llamado justo antes de que la aplicación sea destruida, incluso si el dispositivo cuenta con poca memora es posible que este método se llame antes de querer salir de la aplicación.

Destruida se alcanza este estado después de haber llamado al método *finish()* o incluso que el mismo sistema detuvo el proceso [29].

Las aplicaciones Android pueden ser programadas en distintas IDE(Integrated Development Environment - Entorno de Desarrollo Integrado), siempre y cuando se puedan añadir los complementos necesarios.

2.5.3.2. Entorno de desarrollo para Android con Eclipse

Eclipse es un programa informático que contiene un grupo de herramientas de programación que son multiplataforma. Originalmente fue desarrollado por IBM para sustituir sus herramientas *VisualAge*, pero ahora lo desarrolla la *Fundación Eclipse,* organización sin ánimos de lucro que fomenta el código abierto.

SDK (*software development kit*) de Android proporciona las bibliotecas API y todas las herramientas necesarias para poder realizar, probar e incluso depurar aplicaciones Android.

En las imágenes siguientes se muestra la forma para configurar Eclipse y poder desarrollar aplicaciones Android:

Primero se necesita descargar Eclipse de la página *http://www.eclipse.org/downloads/*, se puede utilizar cualquier versión de Android superior a la 3.3.1, descargue la versión *Eclipse IDE for Java Developers* es una opción similar a la de la Figura 2.8. Una vez finalizada la descarga, descomprima la carpeta en una ubicación permanente.

Después se tiene que descargar el complemento SDK de la página

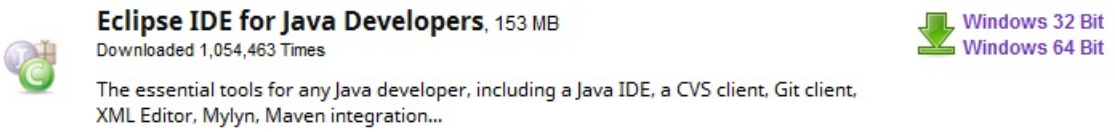


Figura 2.8: Opción de descarga Eclipse IDE for Java Developers.

http://developer.Android.com/intl/es/sdk/index.html ver Figura 2.9 cuando termine la descarga instale el archivo, abrirá una ventana donde mostrará una lista con componentes disponibles, seleccione Accept All y de click en Install, vea la Figura 2.10.Todos los complementos se van a descargar y almacenar en el directorio de instalación del SDK. Seleccione las versiones que va a utilizar, el lenguaje pide la versión más reciente, en éste caso es la Android 4.4.2, y la versión para la que va a compilar la aplicación. La instalación de las demás versiones es opcional pero entre más versiones estén seleccionadas, la descarga e instalación tardará más tiempo (alrededor de 2 horas por versión seleccionada).

Figura 2.9: Opción de descarga SDK.

Una vez que terminaron de instalarse los paquetes se puede proceder a instalar el complemento en Eclipse, para lo cual es necesario abrirlo ejecutando el archivo Eclipse.exe. Cuando esté abierto Eclipse en el menú Help está una opción *Install New Software*, se da click en esa opción, aparecerá una ventana, se da click en el botón *add*, se llenan los campos de nombre y dirección con los datos Android Development Tools y *https://dl-ssl.google.com/Android/eclipse/* como se muestra en la Figura 2.11. Se da click en *ok*, regresará a la ventana anterior, escriba la palabra Android y autocomplete el cuadro con *Android Development Tools* presione la tecla *enter* y en las casillas de

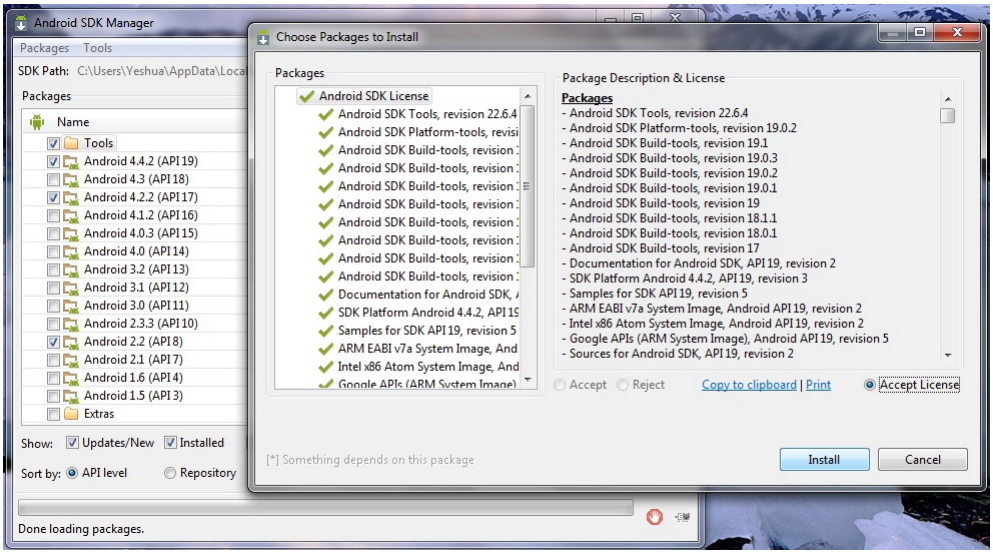


Figura 2.10: Ventanas de SDK Manager.

abajo seleccione *Developer Tools* como se muesta en la Figura 2.12, presione el botón de *next* y nuevamente el botón de *next* en la nueva ventana, seleccione la opción de aceptar los términos y de click en el botón de *Finish*. Una vez terminada la instalación reinicie Eclipse y eso es todo en la parte de la configuración.

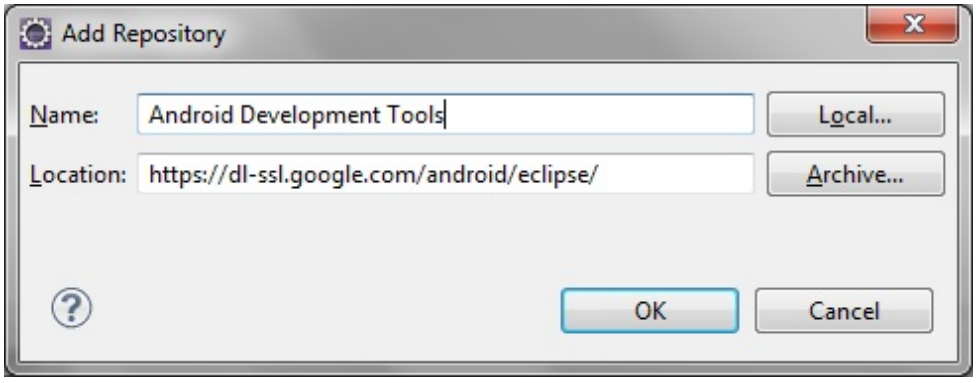


Figura 2.11: Ventana para añadir el complemento de Android a eclipse.

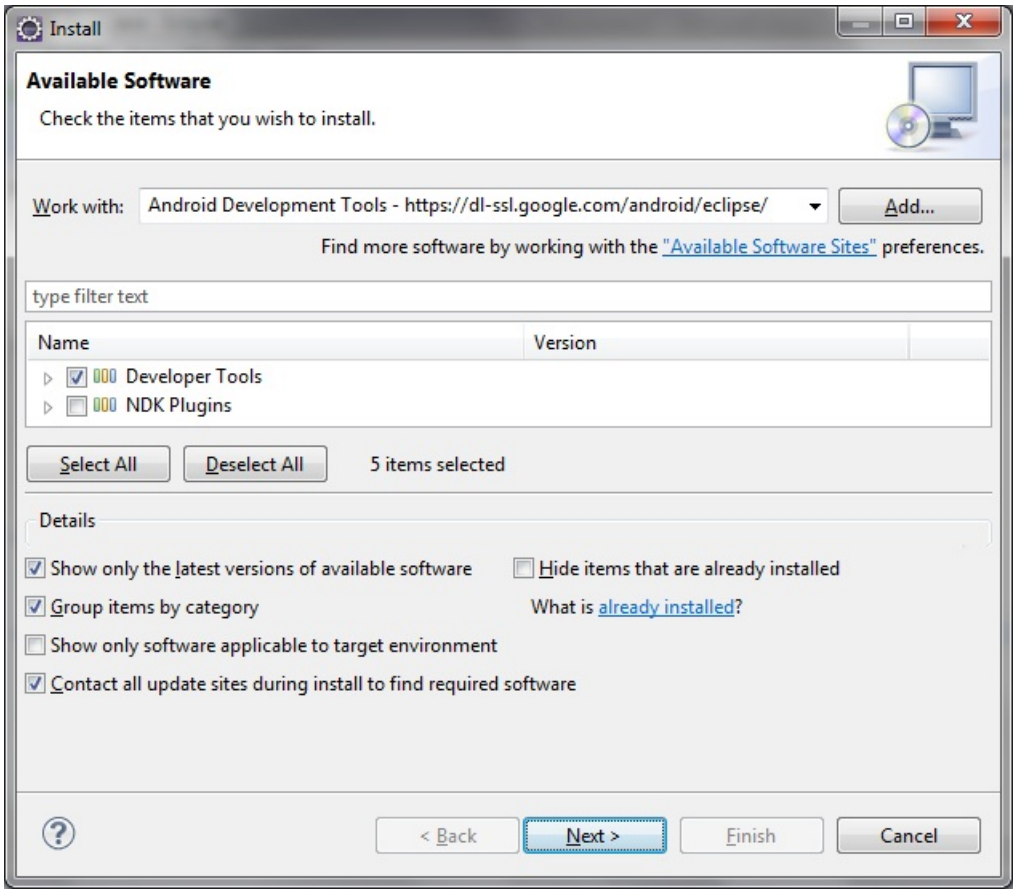


Figura 2.12: Ventana para selecciona la casilla de Developer Tools.

Capítulo 3

Diseño de SIGENIU

El Sistema de Generación de Información Única (SIGENIU) es una aplicación que utiliza la biometría dinámica como principal herramienta para facilitar el estudio del individuo de acuerdo a su comportamiento. Tiene como finalidad obtener un vector único de características que posteriormente puede ser utilizado en múltiples tareas como por ejemplo ser la semilla de una función generadora de números pseudoaleatorios. En este capítulo se presenta el diseño de SIGENIU utilizando los diagramas de casos de uso, de clase, de interacción y de actividades correspondientes al Lenguaje Unificado de Modelado (UML) [8, 7].

3.1. Funcionalidad

El objetivo del sistema es generar un vector de características para cada intento de solución del usuario, su comportamiento garantiza la unicidad de la información generada. La herramienta utilizada para tal propósito es la biometría dinámica de tecleo con el uso de un captcha y la medición del tiempo para cada tecla oprimida.

Un **CAPTCHA** (*Completely Automated Public Turing test to tell Computers and Humans Apart* o *Prueba de Turing completamente automática y pública para diferenciar computadoras de humanos* [11] es una aplicación controlada por una máquina, y su función principal es comprobar que el usuario es un humano y no una máquina, en este caso haciendo referencia a un virus, puesto que un programa simple no tiene la

capacidad de identificar símbolos distorsionados.

El uso del captcha ofrece una ventaja para el sistema ya que es sencillo de entender por parte del usuario, además, de que es muy útil para generar un vector de características.

El vector contiene la información de todas las pulsaciones en el teclado, por ejemplo, el tiempo en milisegundos de cada pulsación o el número de veces que el usuario pulsa una tecla. El promedio de los tiempos, el tiempo mayor, y el tiempo menor entre cada pulsación puede ser calculado a través de la información contenida en el vector de características, datos que pueden ser utilizados como semilla para generar un número pseudoaleatorio o para medir el comportamiento de un usuario e incluso autenticar a un usuario.

El funcionamiento general del sistema debe cumplir con los siguientes requerimientos:

1. El usuario inicia la aplicación
2. El usuario activa y resuelve el captcha
3. El sistema contiene un contador de tiempo
4. El sistema almacena el tiempo requerido por el usuario en cada pulsación
5. El sistema verifica que el usuario haya resuelto el captcha correctamente
6. El sistema genera un vector de caracteristicas a partir de los tiempos obtenidos del usuario
7. El sistema muestra una gráfica del vector de características
8. El usuario termina la aplicación

En las siguientes secciones se presenta el análisis de SIGENIU utilizando el Lenguaje Unificado de Modelado (UML), a través de diagramas de casos de uso, diagrama de clases, diagramas de interacción y diagramas de actividades.

3.2. Casos de uso

La aplicación permite a uno o varios usuarios realizar un análisis de los tiempos que obtuvieron durante la interacción con SIGENIU. La idea principal es obtener información del usuario en su primer interacción con el sistema, sin embargo, un tratamiento de la información resultante de varias interacciones puede ser identificar al usuario con el entrenamiento de su comportamiento.

El diagrama de caso general de SIGENIU se muestra en la Figura 5.1 está compuesto de cinco casos de uso.

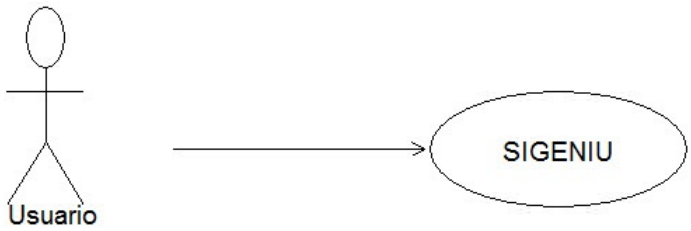


Figura 3.1: Caso de uso general de SIGENIU.

El diagrama de casos de uso detallado, Figura 5.2 está compuesto de cinco casos de uso y un actor.

El caso de uso *generación del captcha* es la acción primordial para el funcionamiento correcto del sistema, genera un captcha pseudoaleatorio que activa el usuario.

El caso de uso *solución del captcha* interactúa con el usuario y permite obtener los tiempos de pulsación del usuario al resolver el captcha.

El caso de uso *almacenamiento de tiempo* tiene como objetivo almacenar en memoria temporal del dispositivo la información obtenida del usuario.

El caso de uso *generación de vector de características* transforma la información de la memoria temporal a un arreglo de datos.

Por último, el caso de uso *impresión gráfica* muestra la información procesada del vector de características.

A continuación se listan las tablas de descripción detallada para cada uno de ellos.

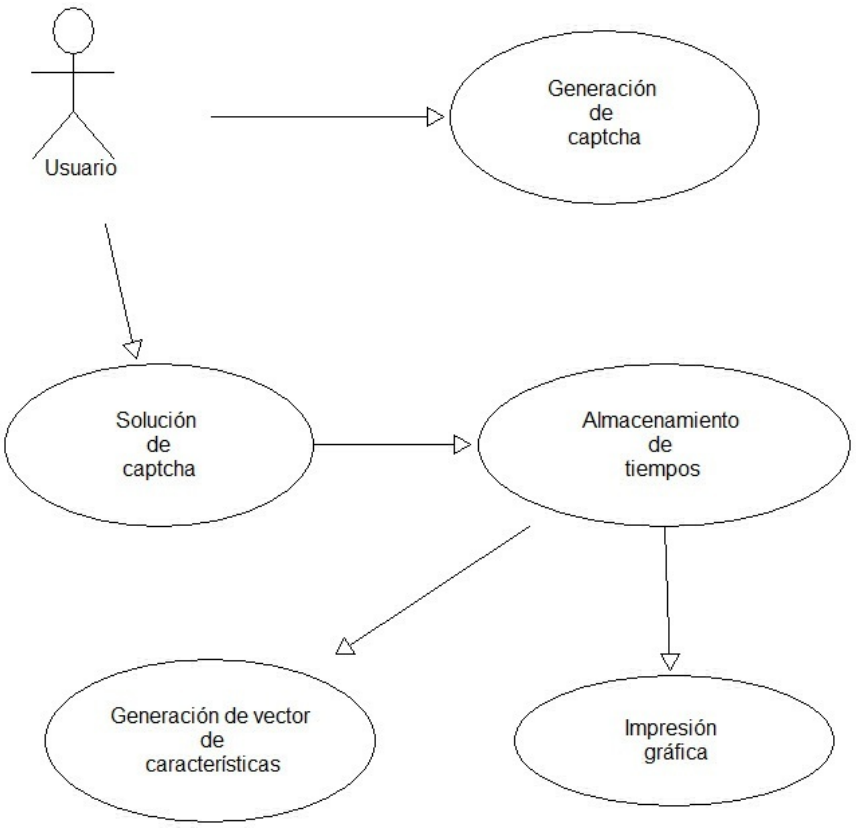


Figura 3.2: Caso de uso SIGENIU detallado.

3.3. Diagrama de clases

En la Figura 3.3 se muestra el diagrama de clases general de SIGENIU, el cual, está compuesto por siete clases. La interacción entre las clases se muestra en el diagrama de clases detallado en la Figura 3.4.

La clase *Bienvenida* es una animación con tiempo, muestra una ventana por 2500 milisegundos (*ms*) y después hace una llamada a la clase *MainActivity* que es la que muestra el captcha. Ésta clase crea un objeto de las clases *UserActivity* y *Componente*. La clase *UserActivity* es la encargada de crear el vector de tiempos en la memoria temporal del dispositivo, la clase componente es la que genera los elementos del captcha (las letras y la distorsión de las mismas). *MainActivity* además de mostrar los elementos mencionados realiza la validación de la información así como la llamada a la clase *ShowKeyTime*.

ShowKeyTime contiene elementos, uno de ellos es la clase *ShTiK* que en pocas

Tabla 3.1: Descripción del caso de uso detallado Generación de Captcha.

Nombre	Generación de captcha.
Actores	Usuario.
Descripción	Cuando el usuario presiona el componente de captcha este se activa y se genera de manera pseudoaleatoria la información.
Disparador	El usuario debe presionar el componente para ser activado.
Precondiciones	1. El usuario debe haber iniciado la aplicación. 2. El usuario debe haber pasado por la pantalla de bienvenida de la aplicación y estar en la pantalla de captcha.
Postcondiciones	1. El captcha está impreso en el componente. 2. Se activa el área para resolverlo. 3. Se activa un contador de tiempo.
Flujo normal	1. Iniciar la aplicación. 2. Se muestra una la ventana principal en estado de inicio. 3. El componente muestra un mensaje que dice “Click aquí”.
Flujo alternativo	1. Se presiona el captcha más de una vez. 1.1. Se genera un captcha diferente por cada vez que es presionado el captcha. 2. Se presiona el botón de atrás. 2.1. La aplicación termina. 3. El usuario presiona el botón de inicio en el dispositivo. 3.1. La aplicación se queda en espera hasta que se vuelva a iniciar.
Excepciones	1. El usuario intenta presionar en el área donde se resuelve el capcha sin haberlo activado. 1.1. El sistema envía un mensaje de error. 2. Se presiona el botón de siguiente sin haber inicializado el captcha. 2.1. El sistema envía un mensaje de error y se señala el componente.
Prioridad	Alta
Frecuencia de uso	Alta

palabras recibe el vector de características y realiza la gráfica correspondiente. Por último esta la clase *MoreInformation* que al igual que la clase *ShTiK* recibe el vector

Tabla 3.2: Descripción del caso de uso detallado Solución de Captcha.

Nombre	Solución del captcha.
Actores	Usuario.
Descripción	El usuario intenta resolver el captcha en el área destinada a ello, de manera que tiene que escribir cada letra que aparezca en el captcha.
Disparador	El usuario debe presionar el componente del captcha.
Precondiciones	1. El usuario debe estar en la ventana correcta. 2. El usuario debe haber de haber presionado por lo menos una vez el área del captcha.
Postcondiciones	1. El captcha está resuelto. 2. Se detiene el contador de tiempo. 3. Se presiona el botón de siguiente.
Flujo normal	1. Se inicia el intento de solución del captcha. 2. El usuario presiona cada letra del captcha. 3. Se presiona el botón de siguiente. 4. Aparece una nueva ventana.
Flujo alternativo	1. El captcha es incorrecto 1.1. Se limpia el área de solución. 1.2. Aparece un mensaje informando que no es correcta la solución. 2. Se presiona el botón de atrás. 2.1. La aplicación termina. 3. El usuario presiona el botón de inicio en el dispositivo. 3.1. La aplicación se queda en espera hasta que se vuelva a iniciar.
Excepciones	1. El usuario presiona el botón de siguiente sin haber terminado de resolver el captcha. 1.1 El sistema envía un mensaje de textos diferentes.
Prioridad	Alta.
Frecuencia de uso	Alta.

Tabla 3.3: Descripción del caso de uso detallado Almacenamiento de tiempos.

Nombre	Almacenamiento de tiempos.
Actores	Solución del captcha.
Descripción	En cada presión de teclado se guarda en la memoria temporal del dispositivo un nuevo dato con el tiempo que tardo en presionar la tecla.
Disparador	El usuario presiona una tecla en el area de solución.
Precondiciones	El usuario tiene que haber activado el area de solución, presionando por lo menos una vez el área del captcha.
Postcondiciones	1. El arreglo contiene el tiempo de todas las presiones de teclado.
Flujo normal	1. Se añade al arreglo un nuevo dato cada vez que el usuario presiona una tecla en el área de solución. 2. El usuario termina de escribir las letras del captcha. 3. El arreglo es enviado a una nueva ventana para ser procesado.
Flujo alternativo	1. El captcha continua creciendo hasta que el usuario deja de escribir. 2. Se presiona el botón de atrás. 2.1. La aplicación termina. 3. El usuario presiona el botón de inicio en el dispositivo. 3.1. La aplicación se queda en espera hasta que se vuelva a iniciar.
Excepciones	-
Prioridad	Alta
Frecuencia de uso	Alta
Reglas	Los datos del arreglo deben de ser únicamente números.

como parámetro, la diferencia es que esta clase realiza algunas operaciones matemáticas para dar como resultado otras características de la información.

Tabla 3.4: Descripción del caso de uso detallado Generación del Vector de Características.

Nombre	Generación del Vector de Características.
Actores	Almacenamiento de tiempos.
Descripción	Cuando el usuario presiona el botón de siguiente, se crea el vector que contiene toda la información de la interacción del usuario con el sistema.
Disparador	El captcha solucionado de forma correcta.
Precondiciones	1. El usuario tiene que haber resuelto correctamente el captcha. 2. El usuario tiene que haber presionado el botón de siguiente.
Postcondiciones	Obtención de información única del usuario.
Flujo normal	1. Se crea el vector de características en la memoria del dispositivo. 2. Se almacena el vector de características en la memoria extraíble del dispositivo.
Flujo alternativo	1. Se presiona el botón de atrás. 1.1. La aplicación termina. 2. El usuario presiona el botón de inicio en el dispositivo. 2.1 La aplicación se queda en espera hasta que se vuelva a iniciar.
Excepciones	1. El teléfono se queda sin batería. 1.1 El archivo puede no escribirse de forma correcta.
Prioridad	Alta.
Frecuencia de uso	Alta.

3.4. Diagramas de secuencia

En la Figura 3.5 se puede observar la interacción del usuario con el sistema. Como ya se mencionó anteriormente, la clase *MainActivity* muestra el componente, el usuario

Tabla 3.5: Descripción del caso de uso detallado Impresión Gráfica.

Nombre	Impresión Gráfica.
Actores	Generación del Vector de Características.
Descripción	Es la ventana en la que se muestra la gráfica de todos los tiempos de las pulsaciones de teclado para la solución.
Disparador	La creación del vector de características para ser leído al momento de realizar la gráfica.
Precondiciones	1. El usuario debe haber presionado el botón de siguiente para que el sistema realice las acciones correspondientes. 2. El sistema debe de haber creado el archivo. 3. El sistema tiene que haber cargado en memoria temporal la información.
Postcondiciones	1. La gráfica se realizo correctamente. 2. Se muestra la gráfica en pantalla.
Flujo normal	1. Se dibuja la gráfica correspondiente al número de intervalos. 2. Se grafican uno a uno los tiempos del vector. 3. Se escribe el valor del tiempo correspondiente en la parte superior de cada barra.
Flujo alternativo	1. Se presiona el botón de atrás. 1.1. El sistema regresa a la ventana principal. 2. El usuario presiona el botón de inicio en el dispositivo. 2.1 La aplicación se queda en espera hasta que se vuelva a iniciar.
Excepciones	1. El usuario apaga el teléfono. 1.1 El sistema termina.
Prioridad	Alta.
Frecuencia de uso	Alta.

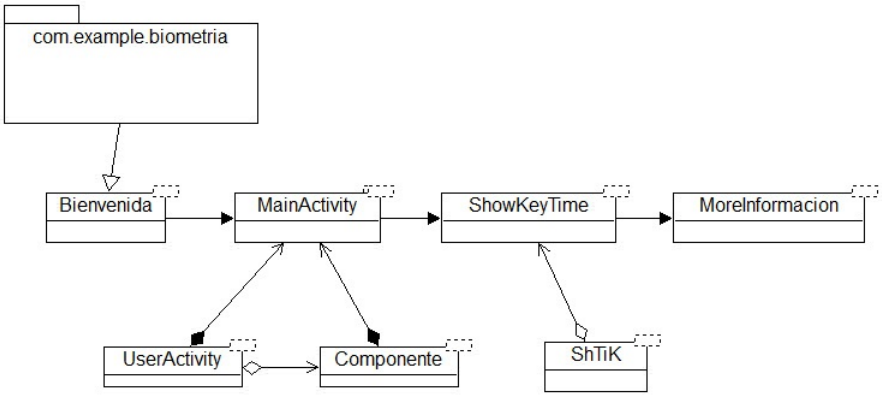


Figura 3.3: Relaciones entre clases.

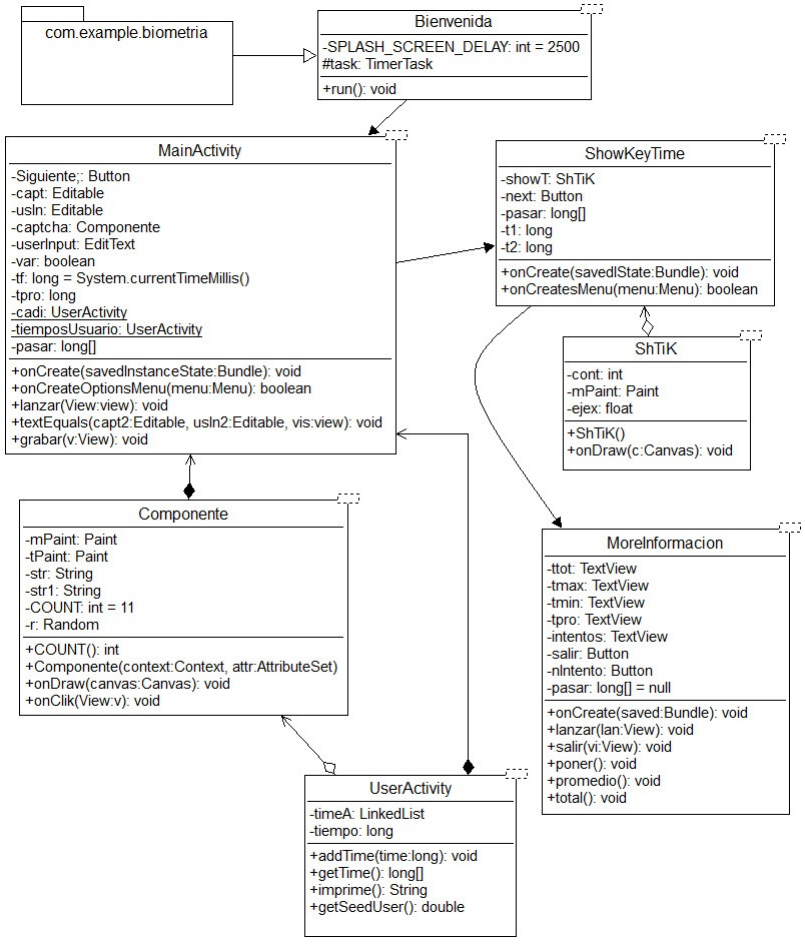


Figura 3.4: Diagrama de clases detallado.

activa el generador y se crea un captcha. La Figura 3.6 describe las acciones que realiza el sistema cuando el usuario presiona una tecla para resolver el captcha y las posibles

acciones cuando verifica la información.

En la figura 3.7 se muestran los pasos que realiza el sistema para almacenar los tiempos obtenidos del usuario, el mecanismo para depositar los tiempos en un vector de características se describe la Figura 3.8. Por último la Figura 3.9 muestra la forma en la que el sistema crea la gráfica representando todos los tiempos obtenidos.

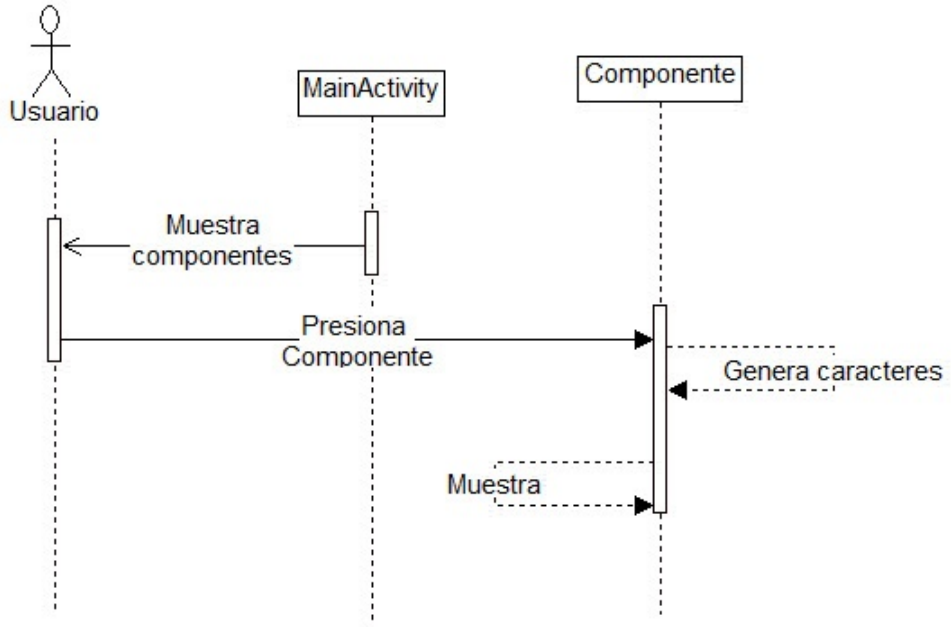


Figura 3.5: Diagrama de secuencia de la generación del captcha.

3.5. Diagrama de actividades

En un bosquejo general de la funcionalidad del sistema y la interacción de todos los casos de uso, la Figura 3.10 muestra el diagrama de actividades, que contiene el flujo de cada proceso de SIGENIU. El sistema requiere al menos siete actividades secuenciales y una de forma concurrente al requerir almacenar los tiempos del usuario según su comportamiento.

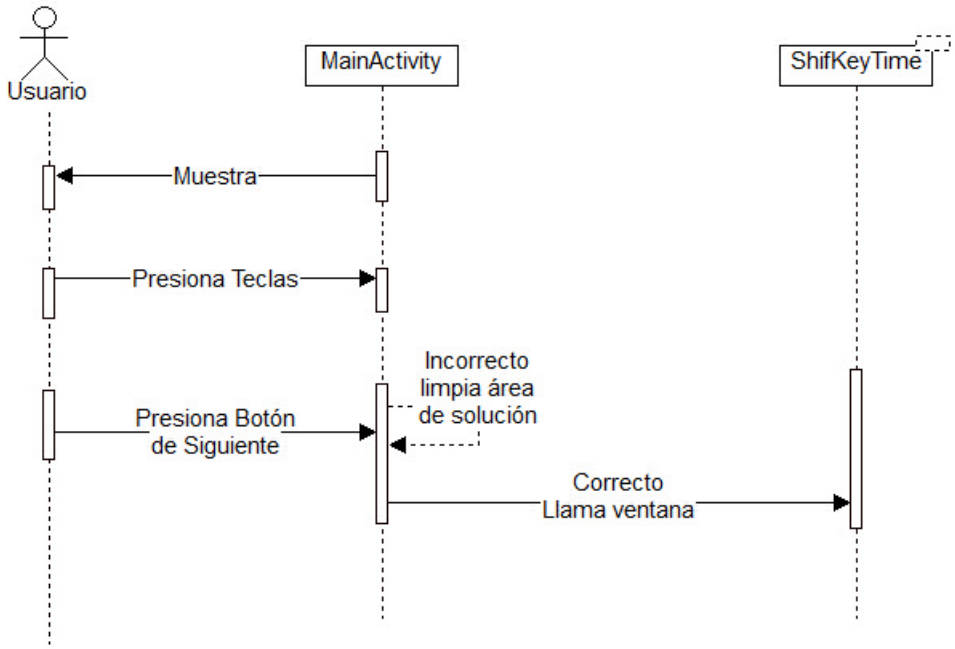


Figura 3.6: Diagrama de secuencia de la solución del captcha.

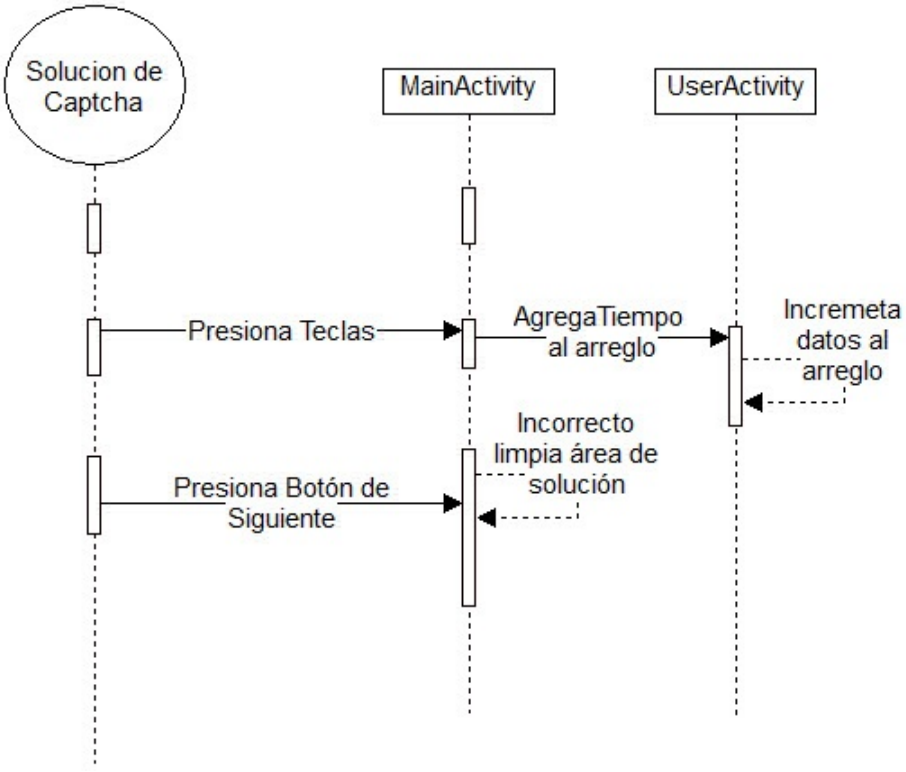


Figura 3.7: Diagrama de secuencia del almacenamiento de tiempos.

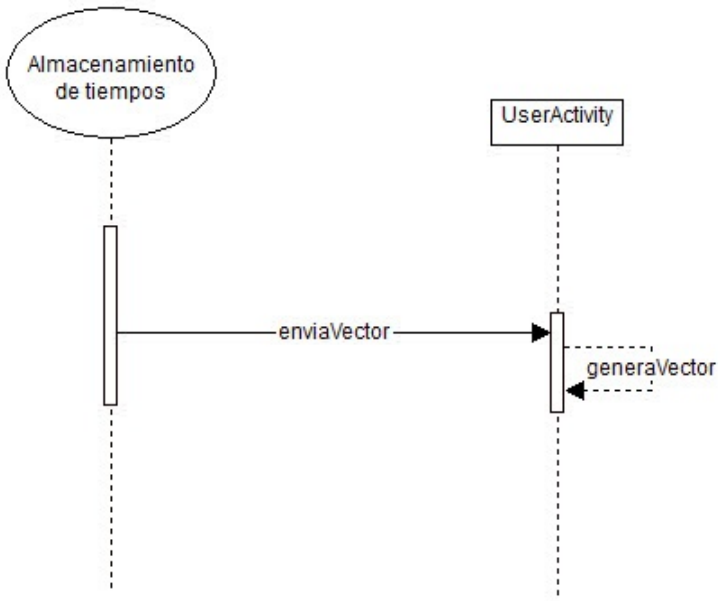


Figura 3.8: Diagrama de secuencia de la generación del vector de características.

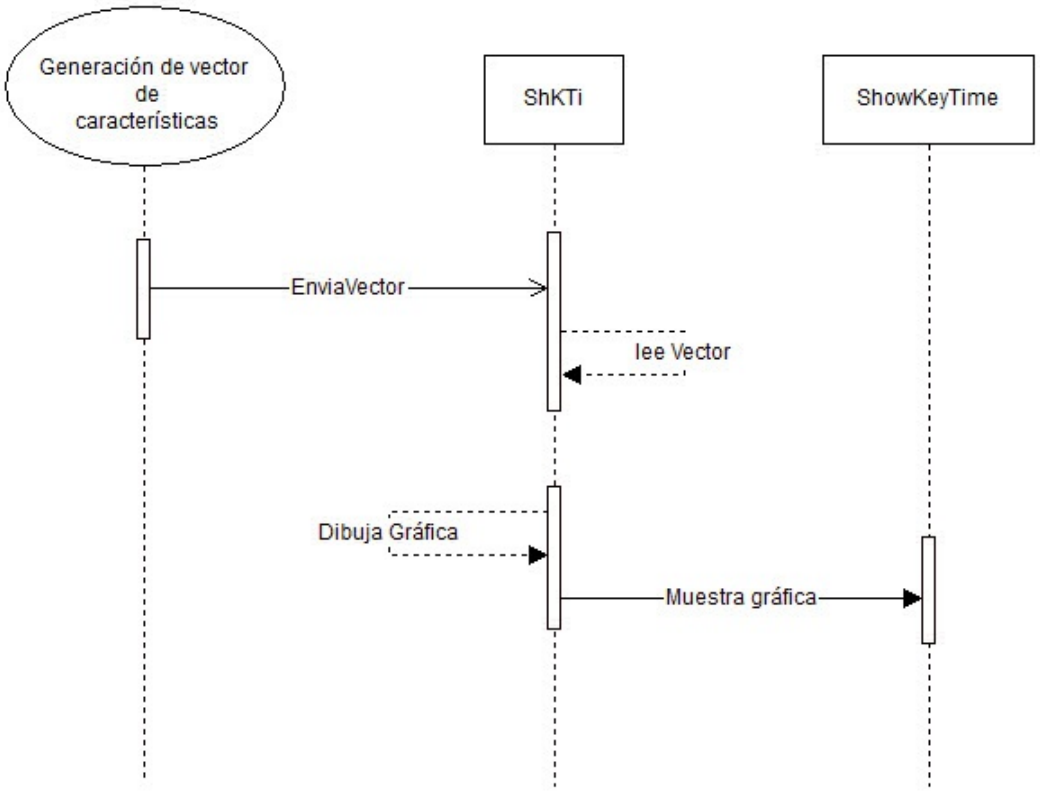


Figura 3.9: Diagrama de secuencia de la impresión gráfica.

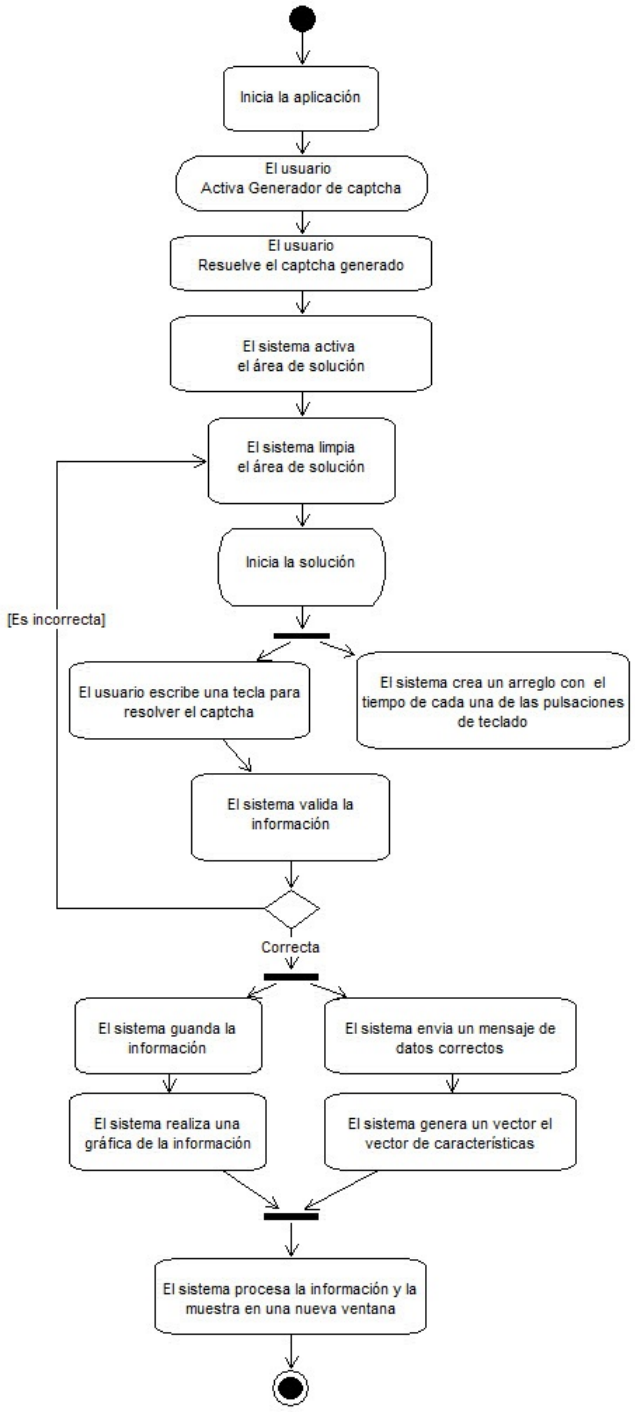


Figura 3.10: Diagrama de actividades de SIGENIU.

Capítulo 4

Implementación de SIGENIU

En este capítulo se presenta el detalle de la implementación de SIGENIU, describiendo la infraestructura utilizada y algunos códigos, que se consideran más relevantes en el funcionamiento del sistema propuesto.

4.1. Infraestructura

SIGENIU fue implementado bajo la plataforma Android compatible con la versión 2.2 en adelante. Las pruebas fueron realizadas en varios teléfonos y una tableta electrónica, de diferentes marcas como Samsung, Motorola, ekt y Tech iPad.

Las características de algunos de los equipos móviles en los que se implementó el sistema son las siguientes:

- Galaxy Mini S5570 Android 2.2, procesador 600 MHz, 280 MB RAM
- Motorola XT.303 Android 2.3.6, procesador de 800Mhz, 256 MB RAM
- EKT KA-5926 Android 4.0.4, procesador 800 MHz, 500MB RAM
- Tech iPad xtab-781+ Android 4.1, procesador 1.5GHz, 1GB RAM

Las características de la PC en la que se desarrollo e implemento son:

- HP NoteBook G42

- Procesador Intel i3, 2.3 GHz
- 3 GB en memoria RAM
- 500 GB en disco duro

4.2. Procesos en general

SIGENIU fue implementado siguiendo el diagrama de flujo general mostrado en la Figura 4.1. Cada proceso sigue las especificaciones mencionadas en los diagramas de caso de uso, interacción y de actividades, de acuerdo con las clases indicadas en el diagrama de clases.

Para hacer más entendible el funcionamiento en general de la implementación, los procesos del diagrama de flujo están en dos colores diferentes de acuerdo a su importancia para el sistema. El primer paso es esencial ya que si no se genera un captcha el siguiente mecanismo no se activa. El siguiente paso es activar el contador de tiempo. Después de solucionar el captcha se verifica que sea correcta la solución, en caso contrario se repite el proceso, mientras tanto el vector de características continúa creciendo. Al presionar el botón de siguiente, se compara el captcha con lo que el usuario escribió.

El sistema presenta pantallas de interacción con el usuario las cuales están descritas a continuación. En la Figura 4.2 se puede visualizar la pantalla de bienvenida, que muestra las credenciales del desarrollador.

La Figura 4.3 muestra la interfaz principal del sistema. El usuario activa el captcha y éste se genera de forma pseudoaleatoria.

El siguiente paso, la Figura 4.4, consiste en resolver el captcha. En ese instante, el contador de tiempo se activa y comienza a almacenar la información para cada tecla que es presionada a través de un *escucha* que detecta los cambios en el texto escrito en el control. Siendo un *escucha* un método de *java* que se encarga de monitoriar el componente, este se añade al elemento que lo requiera.

Cada vez que el usuario introduce un caracter, se mide el tiempo y se almacena. Al finalizar el captcha, se verifica si lo escrito por el usuario es igual a lo mostrado en el

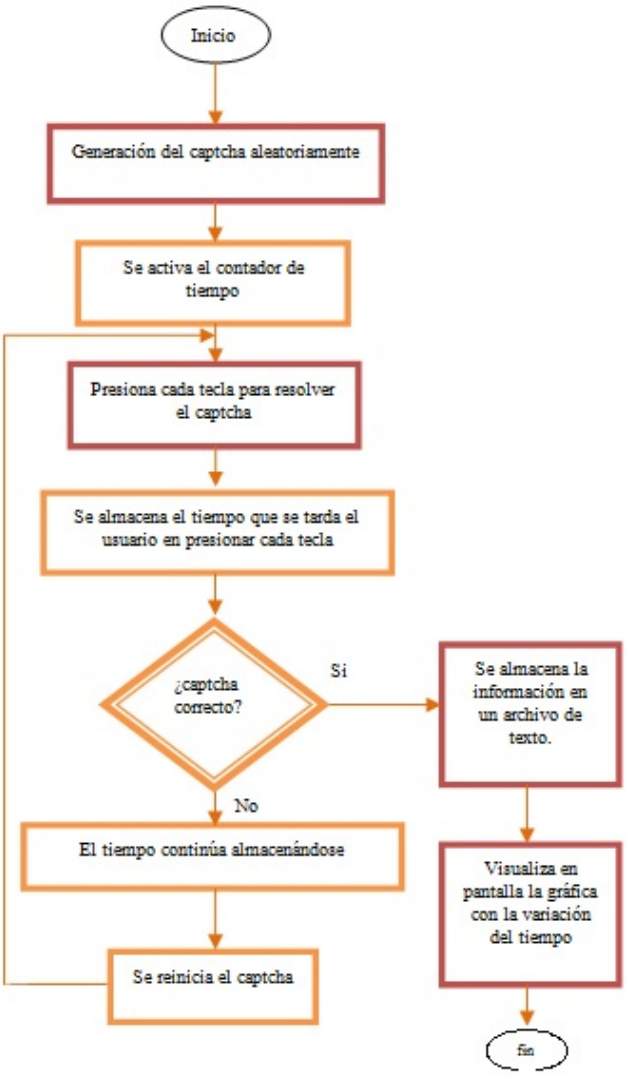


Figura 4.1: Diagrama de flujo del funcionamiento de SIGENIU.

captcha.

Si el usuario tuvo un error, el proceso se reinicia y se realiza el número de veces necesario hasta que el captcha sea resuelto correctamente.

Es importante mencionar que el tiempo es almacenado por cada tecla presionada por el usuario, incluyendo, las teclas de suprimir y enter. Además de que se inicia su almacenamiento al activarse el captcha y se detiene hasta que éste sea resuelto.

Una vez resuelto el captcha de forma correcta, el sistema realiza una gráfica de los tiempos de interacción del usuario con el sistema. La Figura 4.5 muestra una gráfica como ejemplo.

Figura 4.2: Pantalla de bienvenida de la aplicación.

Figura 4.3: Pantalla principal del sistema.

Por último, la Figura 4.6 muestra la información correspondiente al comportamiento del usuario, como el total del tiempo requerido, el número de veces que se tecleo y el promedio de los tiempos mayor, menor y total.

4.2.1. Codificación

Un captcha nuevo es generado cada vez que el usuario pulsa en el componente del mismo. Entonces es necesario realizar una interfaz de *java* llamada *OnClickListener* que hace posible que se pueda hacer un captcha diferente cada vez que se pulsa. Lo anterior se puede observar en el código de la Figura 4.7.

Figura 4.4: Pantalla de solución del captcha.

Figura 4.5: Pantalla de una gráfica de los tiempos.

Cada vez que se genera un captcha, se sobreescribe el método de la interfaz mencionada, misma que genera dos cadenas de letras separadas por un espacio en blanco.

La Figura 4.8 muestra un fragmento de código el cual añade un *escucha* al área de solución, dicho escucha realiza una llamada al método *afterTextChanged* el cual añade un nuevo tiempo al arreglo en memoria temporal.

La Figura 4.9 contiene el código que permite realizar la escritura de los datos en la memoria física del dispositivo, en caso de no encontrar memoria disponible envía un mensaje que indica que los datos no se escribieron. En el caso de no tener problema el sistema envía un mensaje indicando que los datos se escribieron de forma correcta.

Figura 4.6: Pantalla de información extra.

```
public void onClick(View v) {
    java.util.Random r = new java.util.Random();
    r.setSeed(System.currentTimeMillis());
    str = "";
    for (int i = 1; i <= 4; i++) {
        str = str + (char) ('A' + (r.nextInt('Z' - 'A')));
    }
    str1 = str + " ";
    for (int i = 1; i <= 4; i++) {
        str1 = str1 + (char) ('A' + (r.nextInt('Z' - 'A')));
    }
    setText(str1);
}
```

Figura 4.7: Fragmento de código que genera las letras del captcha de forma pseudoaleatoria.

```
userInput.addTextChangedListener(new TextWatcher() {

    @Override
    public void onTextChanged(CharSequence s, int start, int before,
            int count) {
        setTpro(System.currentTimeMillis());
    }

    @Override
    public void beforeTextChanged(CharSequence s, int start, int count,
            int after) {}

    @Override
    public void afterTextChanged(Editable s) {
        getTiemposUsuario().addTimeActivity(
                System.currentTimeMillis() - tf);
        tf = System.currentTimeMillis();
    }
});
```

Figura 4.8: Fragmento de código que añade los tiempos al arreglo en memoria temporal.

La Figura 4.10 muestra el código de los métodos *lanzar* y *grabar*. El método *textEquals* verifica si la solución del captcha es correcta o no, si es correcto realiza una

```
public void grabar(View v) {
    String nomarchivo = "tiempo.txt";
    String contenido = tiemposUsuario.imprime() + "\n-----\n";
    File tarjeta = Environment.getExternalStorageDirectory();
    File file = new File(tarjeta.getAbsolutePath(), nomarchivo);
    java.io.RandomAccessFile fout;
    try {

        fout = new RandomAccessFile(file, "rw");
        fout.seek(fout.length());
        fout.writeChars(contenido);
        Toast.makeText(this, "Los datos fueron grabados correctamente",
                Toast.LENGTH_SHORT).show();
        fout.close();
    } catch (Exception ex) {
        Toast.makeText(this, "Los datos no se escribieron, posiblemente no " +
                "tenga memoria disponible",Toast.LENGTH_SHORT).show();
    }
```

Figura 4.9: Fragmento de código que escribe los datos en memoria de teléfono.

llamada a *lanzar* que es el encargado de crear la ventana en la que se va a mostrar la gráfica de los tiempos y al método de *grabar* encargado de escribir los datos en memoria.

```
public void lanzar(View view) {
    pasar = tiemposUsuario.getTimeActivity();
    Intent i = new Intent(this, ShowKeyTime.class);
    i.setAction(Intent.ACTION_SEND_MULTIPLE);
    i.putExtra("pasar", pasar);
    startActivity(i);
    finish();
}

public void textEquals(Editable capt2, Editable usIn2, View vis) {
    if (capt2.toString().equals(usIn2.toString())) {
        Toast.makeText(this, "Bien hecho!!", Toast.LENGTH_SHORT).show();
        lanzar(vis);
        grabar(vis);
    } else {
        Toast.makeText(this, "Error, Los Textos son diferentes",
                Toast.LENGTH_SHORT).show();
        userInput.setText("");
    }
}
```

Figura 4.10: Fragmento de código de los métodos lanzar y grabar.

La Figura 4.11 realiza el vector de tiempos del usuario.

```
public double getSeedUser() {
    double seed[] = new double[timeActivities.size()];
    java.util.Random r = new java.util.Random();
    r.setSeed(System.currentTimeMillis());
    for (int i = 0; i < timeActivities.size(); i++) {
        long t = ((Long) (timeActivities.get(i))).longValue() >> r
                .nextInt(7);
        seed[i] = (double) t;
        tiempo = tiempo * seed[i];
    }
    return tiempo;
}
```

Figura 4.11: Fragmento de código encargado de crear el vector de tiempos.

La Figura 4.12 muestra un ciclo encargado de crear la gráfica, dibuja una a una las

barras del valor correspondiente.

```
for (int p = 1; p < tiempos.length; p++) {
    x1 = (int) (x0 + porc);
    y1 = (int) (maxYT - ((maxYT / tMax) * tiempos[p]));
    if (tiempos[p] < 140) {
        canvas.drawText(Integer.toString((int) tiempos[p]), x0 + 5,
                y1 - 5, paintT);
    } else {
        canvas.drawText(Integer.toString((int) tiempos[p]), x0 + 5,
                y1 + 15, paintT);
    }
    canvas.drawRect(x0 + 5, yc0, x1, y1, paint);
    canvas.drawText(Integer.toString(p), x1, maxY - 18, paintL);
    x0 = x1;
}
```

Figura 4.12: Fragmento de código encargado de dibujar la gráfica de los tiempos.

Capítulo 5

Pruebas y Resultados

En el presente capítulo se muestran los resultados de la pruebas realizadas en diferentes dispositivos y con diferentes usuarios. Se realizaron alrededor de 90 ejecuciones de las cuales se mostrarán las pantallas y los tiempos de las más representativas.

5.1. Pruebas

Antes de mostrar las pruebas es importante mencionar que el captcha está compuesto de dos bloques de cuatro letras, así, para ser resuelto, requiere que el usuario teclee por lo menos ocho veces.

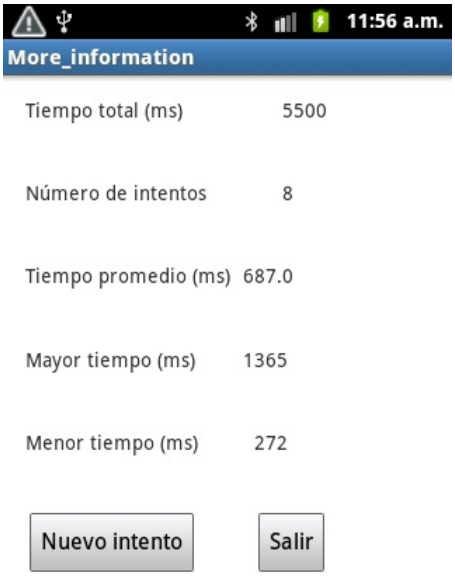
Las pruebas son presentadas de acuerdo al comportamiento del usuario, considerando rapidez, certeza, familiaridad de la infraestructura móvil y familiaridad en el uso del captcha.

A modo de apreciación, se muestra la Figura 5.1, que es la primera prueba realizada. En el inciso(a) se visualiza la pantalla donde se resuelve el captcha. El inciso b presenta una gráfica del comportamiento el usuario, donde se indica el tiempo en milisegundos que tardó el usuario en pulsar cada tecla utilizada. Al mostrarse la leyenda *Los datos fueron grabados correctamente*, significa que la información del usuario se guardó en la memoria del teléfono en un archivo llamado **tiempos.txt**, utilizado posteriormente para generar el vector de características. Por último en el inciso c, se presentan los tiempos obtenidos del comportamiento del usuario, como por ejemplo, el tiempo total,

el número de intentos que es realmente el número de pulsaciones, el tiempo promedio, así como el tiempo de las teclas que fueron presionadas con mayor y menor tiempo.

(a) Solución del captcha.

(b) Gráfica del comportamiento del usuario.



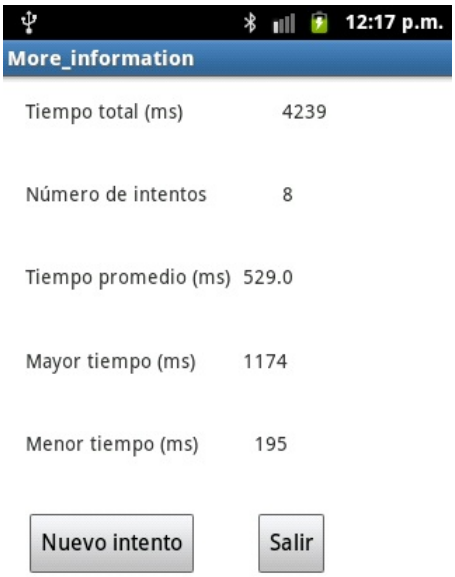
(c) Información extra del usuario.

Figura 5.1: Ejemplo del primer usuario del sistema SIGENIU.

En la Figura 5.2 se presenta el mejor caso, que es cuando el usuario resuelve el captcha de forma certera y rápida, utilizando el mínimo número de pulsaciones y obteniendo 4239 milisegundos.

(a) Solución captcha.

(b) Gráfica de tiempos.



(c) Resultados.

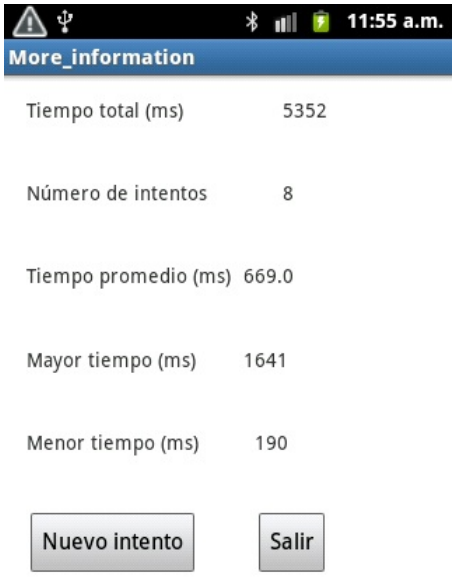
Figura 5.2: Usuario que representa el mejor caso.

En la Figura 5.3 se muestran los resultados para el segundo usuario más rápido. Como puede observarse, aunque sólo ocupó ocho pulsaciones, el tiempo requerido fue

de 5352 milisegundos, mientras que el más rápido fue de 4239.

(a) Solución del captcha.

(b) Gráfica de tiempos.



(c) Resultados.

Figura 5.3: Usuario posible mejor caso.

Cabe señalar, que esta ventana de información en todas las ejecuciones tuvo resultados diferentes, inclusive si lo ejecutó el mismo usuario dos o más veces.

Las Figuras 5.4 (a),(b) y (c) muestran el comportamiento de un usuario que no está familiarizado con el sistema. Al tratar de resolver el captcha tuvo algunos errores, es decir, pulsó en repetidas ocasiones alguna tecla (incluyendo la tecla del espacio), incluso se concluye que resolvió de forma incorrecta el captcha más de una vez y por ello tiene la cantidad de pulsaciones de dos intentos de solución, así mismo, se puede observar que el tiempo de solución es mayor al de otros usuarios, lo que significa que no es certero y eficiente. Se puede considerar a este usuario como el peor caso en el uso del sistema como de los dispositivos móviles.

Las Figuras 5.5 (a),(b) y (c) representan el caso promedio. Los usuarios tardaron en resolver el captcha en un promedio de 6400 milisegundos. El número de pulsaciones también fue ocho pero tardó más en identificar las letras del captcha que el usuario de la Figura anterior. Este tipo de usuario es aquel que primero se cerciora que es la letra correcta y luego la pulsa. Así, asegura de que el captcha sea resuelto correctamente en el primer intento.

(a) Solución del captcha

(b) Gráfica de tiempos.

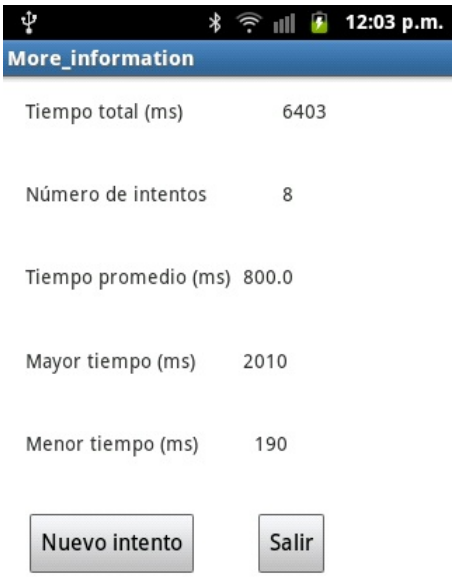
(c) Resultados.

Figura 5.4: Usuario que representa el peor caso.

De acuerdo a uno de los objetivos, el sistema fue diseñado para dispositivos móviles con sistema operativo Android, así como, para computadoras de escritorio. En éste último, el sistema está implementado en Java SE 6. Algunas pruebas realizadas en la computadora personal se muestran en las Figuras 5.6, 5.7 y 5.8. Como puede observarse, también los usuarios tienen errores aunque la interfaz sea más grande y clara. Se

(a) Solución del captcha.

(b) Gráfica de tiempos.



(c) Resultados.

Figura 5.5: Usuario que representa el caso promedio.

solicitaron más datos al usuario con la finalidad de tener un registro de ellos en una base de datos. Si el usuario tuvo problemas para resolver el captcha, el sistema lanza

una ventana de advertencia a diferencia del sistema en el teléfono móvil.

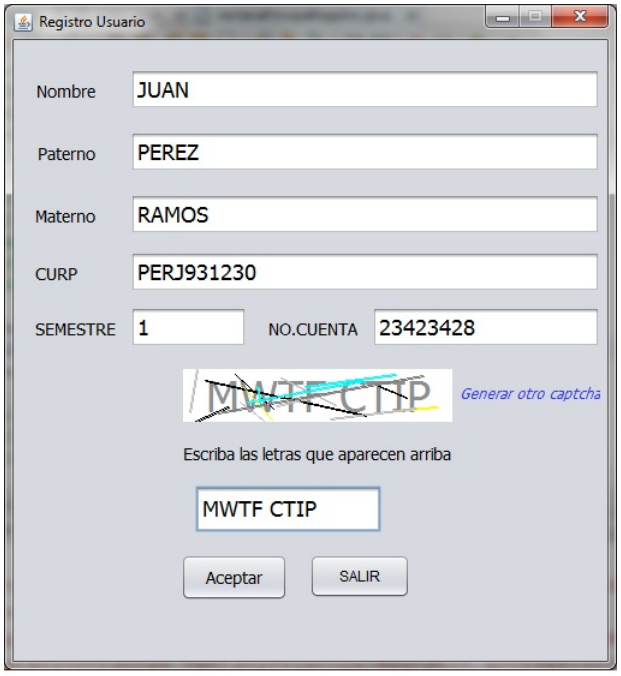


Figura 5.6: Solución del captcha en la PC.

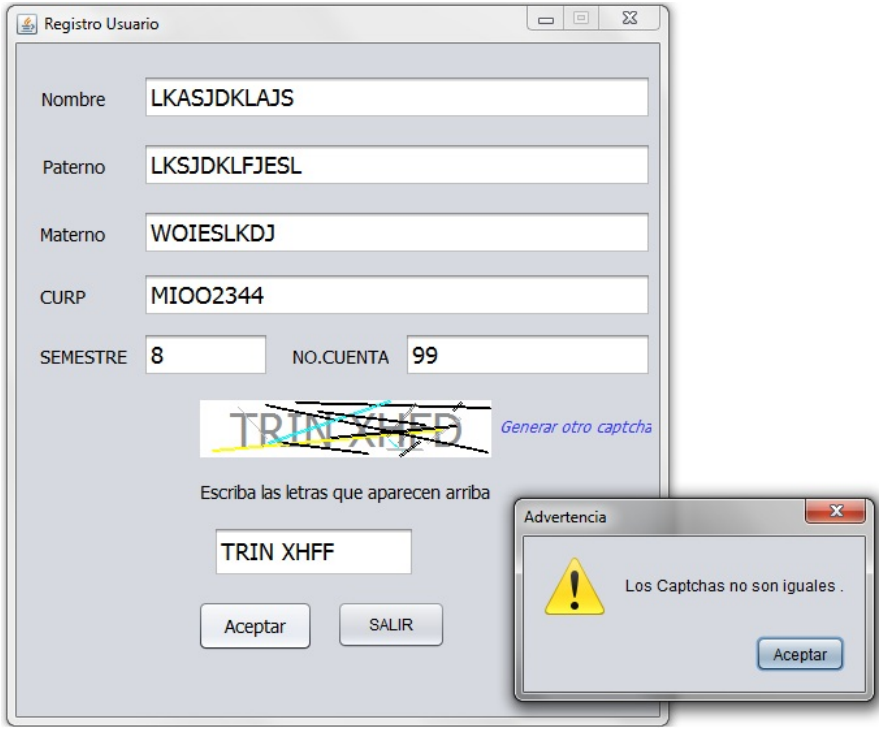


Figura 5.7: Solución del captcha en la PC con dos intentos.

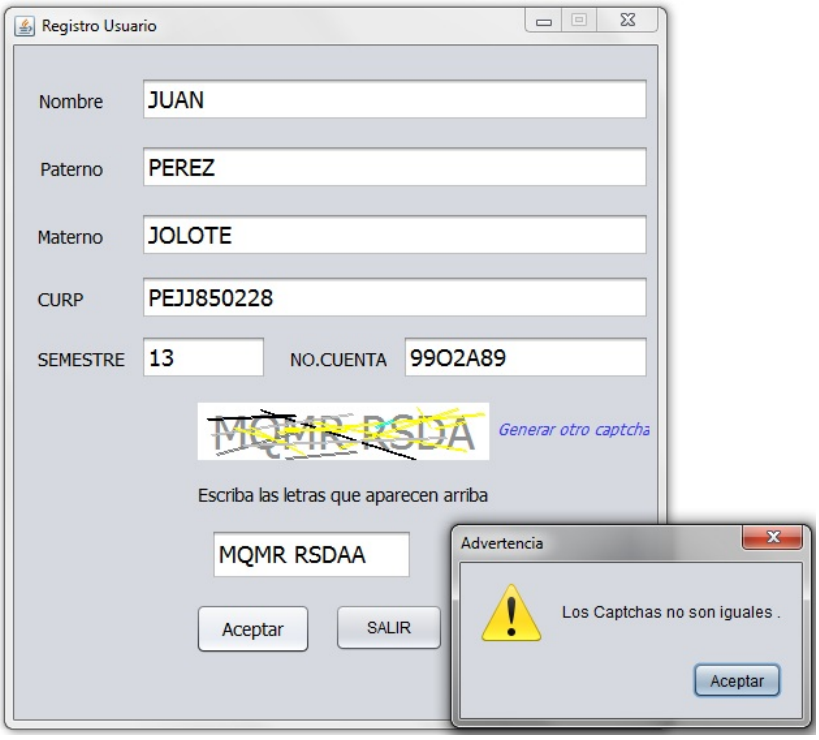


Figura 5.8: Solución del captcha en la PC con más de dos intentos.

5.2. Resultados

Los resultados obtenidos muestran que los usuarios en todos los casos (mejor, promedio y peor) usaron tiempos diferentes al presionar cada tecla. A manera de comprobación, en las Figuras 5.9 a la 5.16 se presentan las gráfica del tiempo de treinta usuarios de las nueve primeras pulsaciones, que en realidad son las mínimas para resolver el captcha.

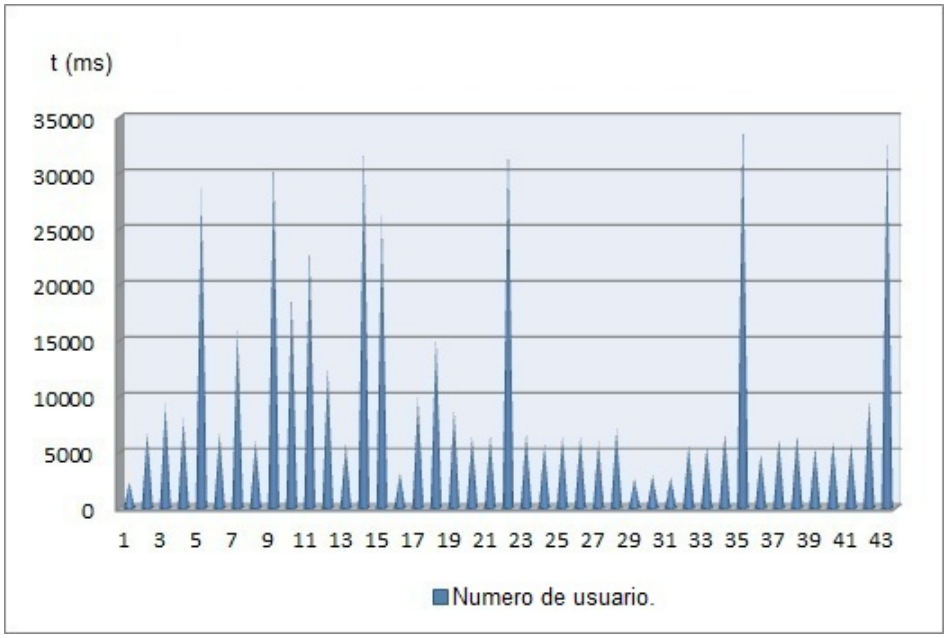


Figura 5.9: Gráfica de los tiempos para la primera pulsación de teclado en cada uno de los usuarios.

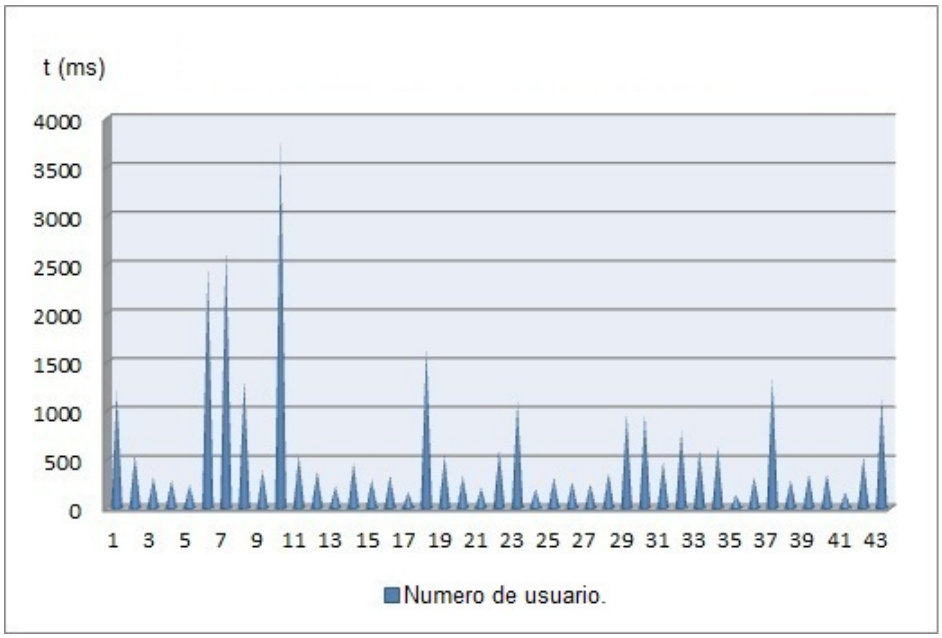


Figura 5.10: Gráfica de los tiempos para la segunda pulsación de teclado en cada uno de los usuarios.

La Figura 5.18 muestra la gráfica con la suma de los tiempos de los treinta usuarios que se eligieron. En algunos casos, como se puede observar, la interacción de algunos

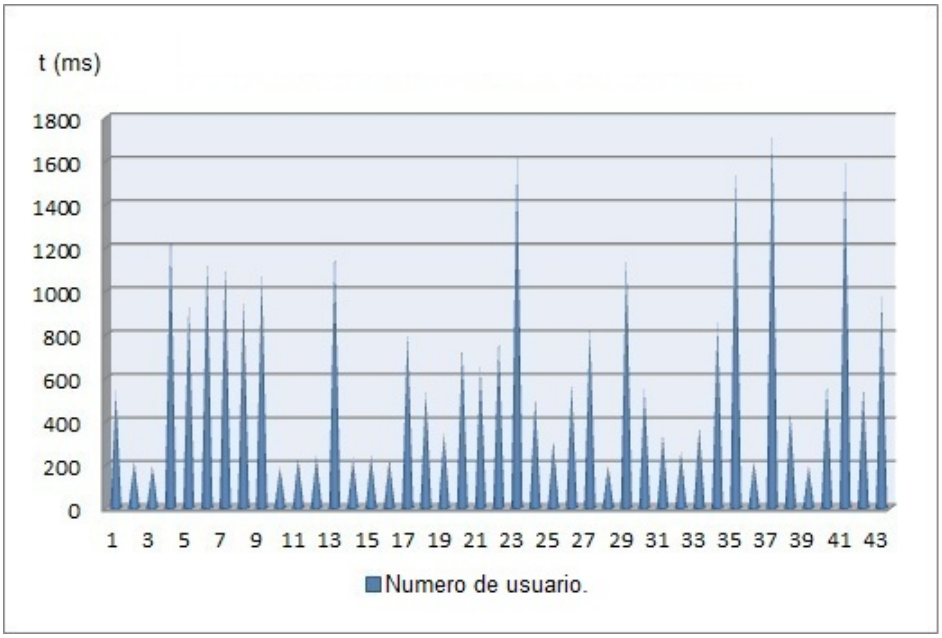


Figura 5.11: Gráfica de los tiempos para la tercer pulsación de teclado en cada uno de los usuarios.

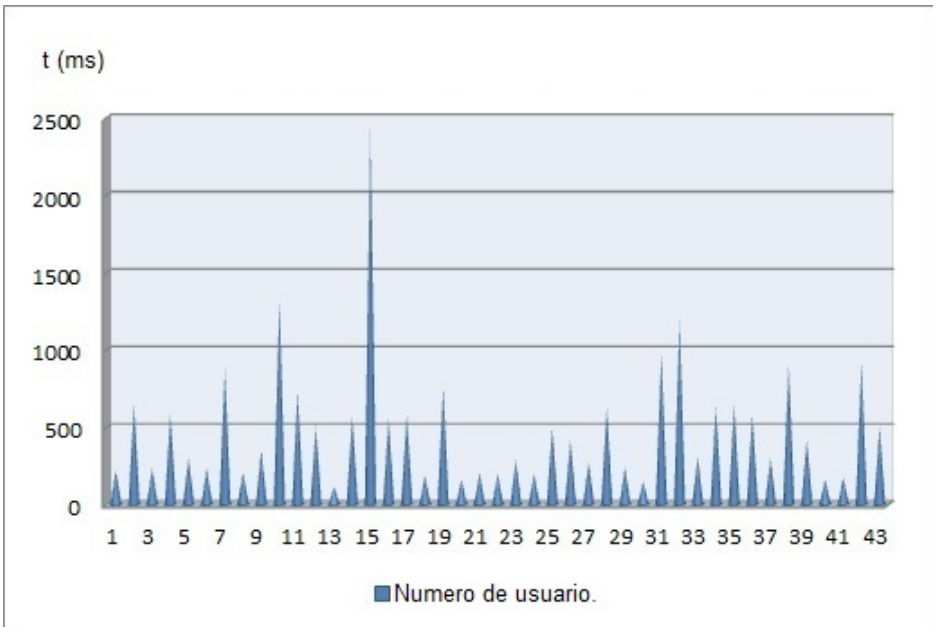


Figura 5.12: Gráfica de los tiempos para la cuarta pulsación de teclado en cada uno de los usuarios.

usuarios esta muy cerca del minuto. Los escenarios posibles para obtener ese tiempo es que el usuario no entendió las instrucciones que se le comunicaron, que no había

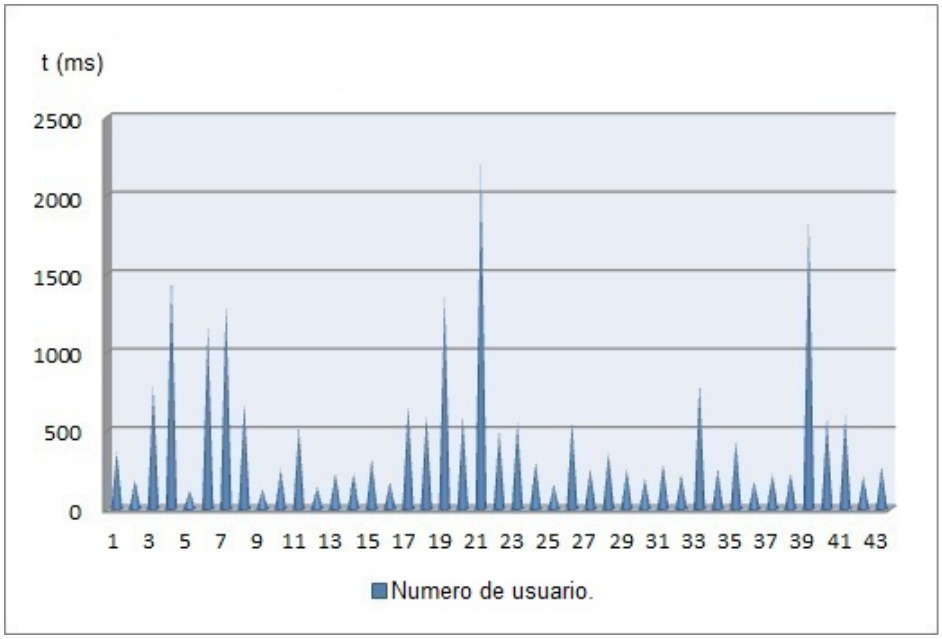


Figura 5.13: Gráfica de los tiempos para la quinta pulsación de teclado en cada uno de los usuarios.

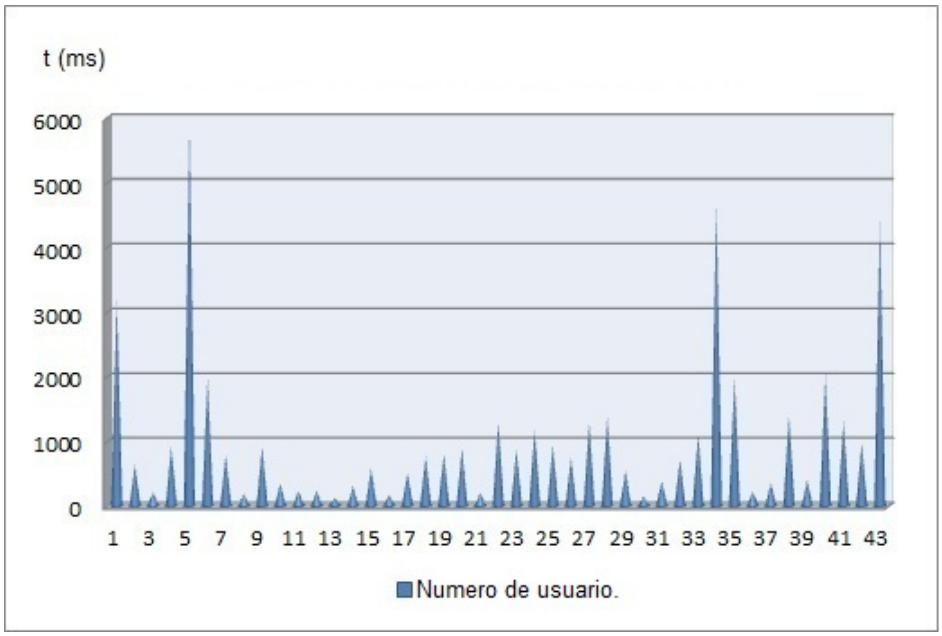


Figura 5.14: Gráfica de los tiempos para la sexta pulsación de teclado en cada uno de los usuarios.

tenido un encuentro con un dispositivo móvil, el captcha era muy confuso para él, o simplemente que no tenía la intención de resolverlo.

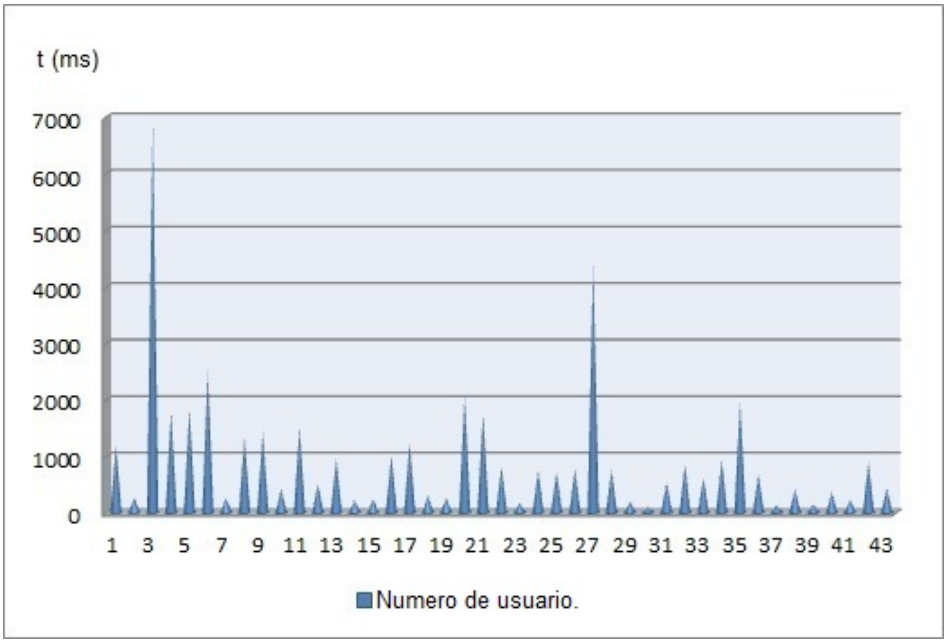


Figura 5.15: Gráfica de los tiempos para la séptima pulsación de teclado en cada uno de los usuarios.

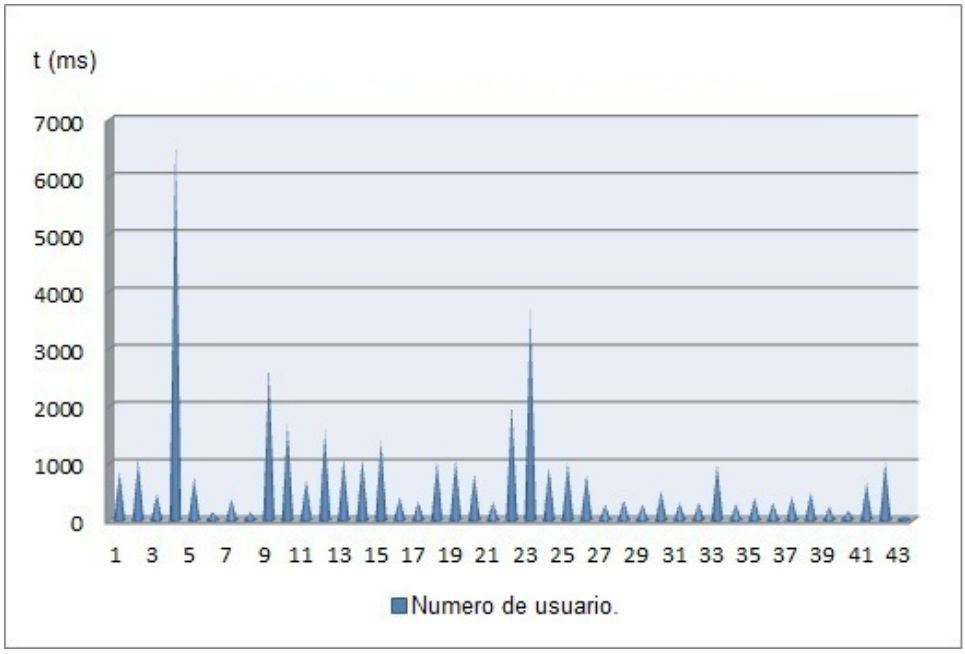


Figura 5.16: Gráfica de los tiempos para la octava pulsación de teclado en cada uno de los usuarios.

La Figura 5.19 muestra la gráfica con el tiempo mínimo que tardaron en pulsar

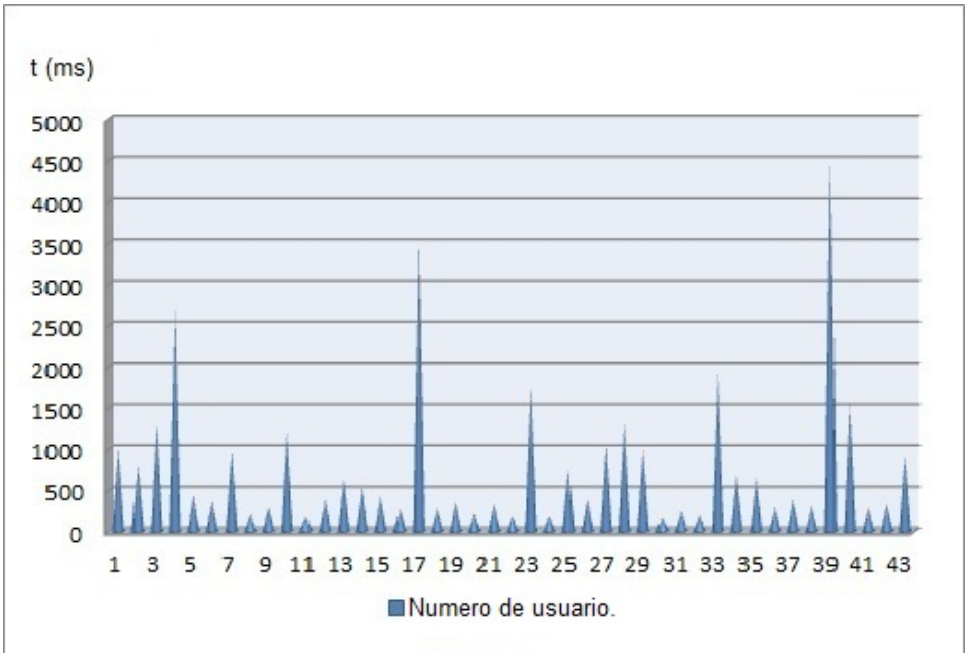


Figura 5.17: Gráfica de los tiempos para la novena pulsación de teclado en cada uno de los usuarios.

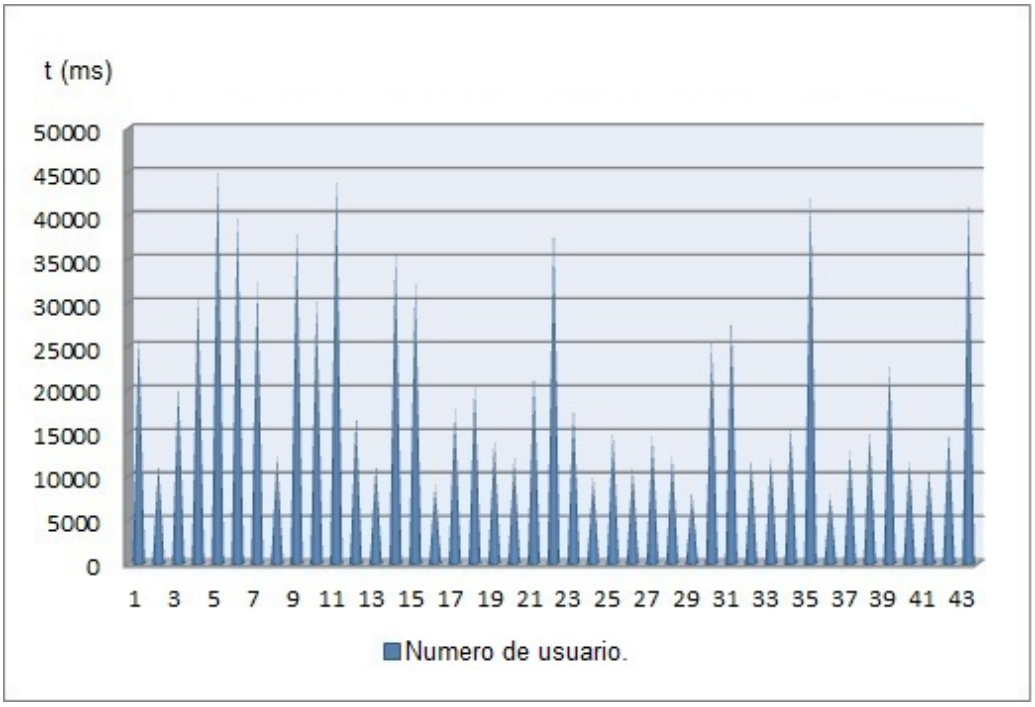


Figura 5.18: Gráfica de la suma de los tiempos en cada uno de los usuarios.

alguna de las letras para treinta usuarios. El tiempo más pequeño reportado fue de 69 milisegundos que normalmente corresponde al cambio entre la tecla espacio y la primera tecla del segundo bloque del captcha.

La Figura 5.20 muestra la gráfica con el tiempo máximo de cada usuario, recordemos que es un captcha y como se genera pseudoaleatoriamente a veces puede ser muy confuso para algunas personas.

La Figura 5.21 muestra la gráfica del tiempo promedio para los mismos treinta usuarios, mientras que la Figura 5.22 representa la media.

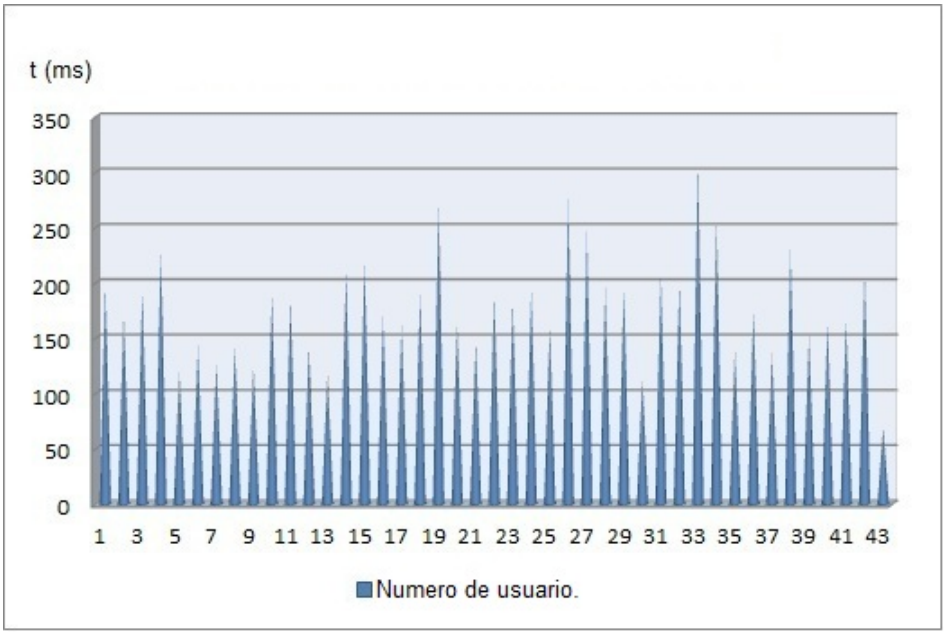


Figura 5.19: Gráfica del tiempo mínimo en cada usuario.

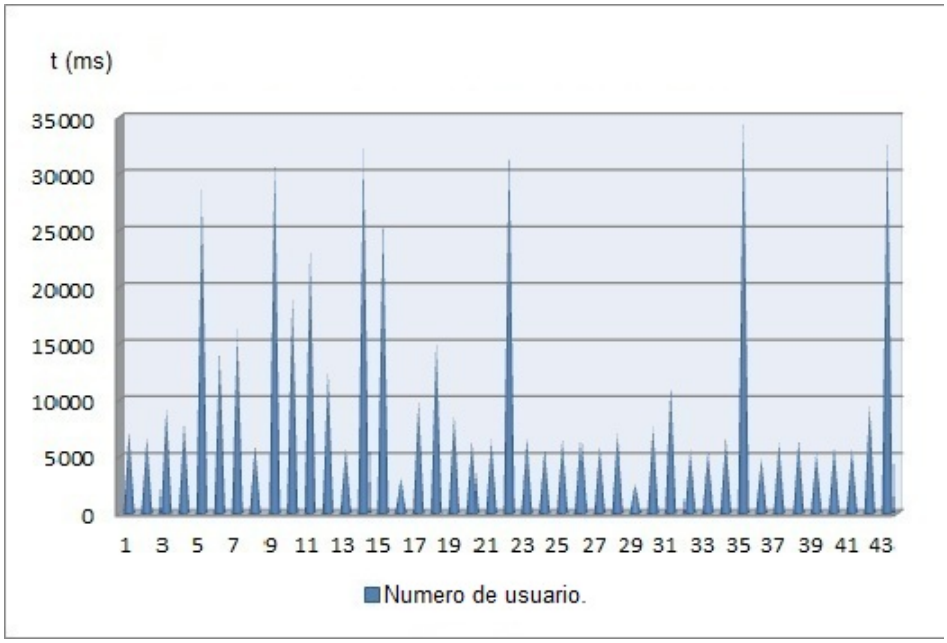


Figura 5.20: Gráfica del tiempo máximo de cada usuario.

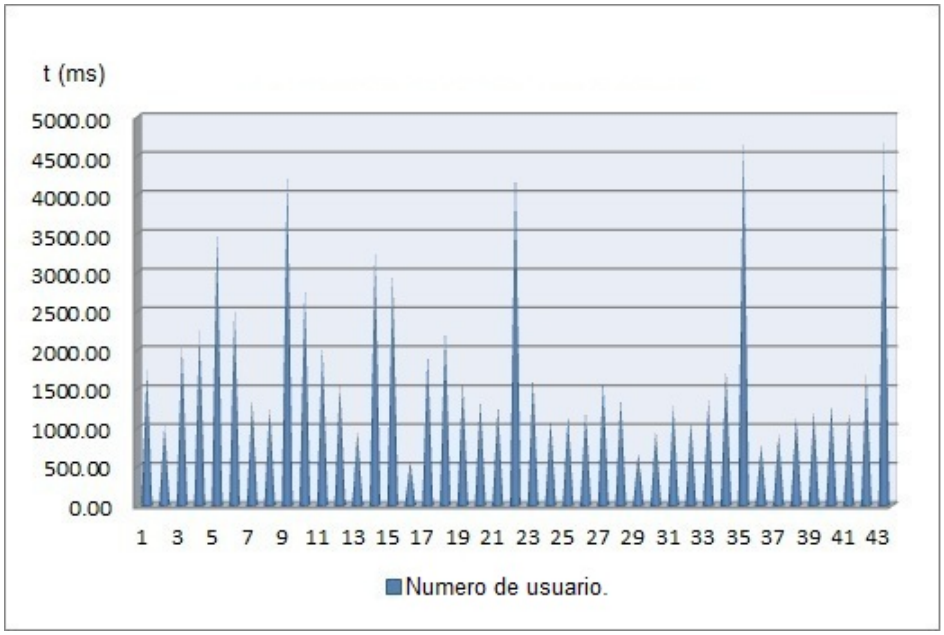


Figura 5.21: Gráfica del promedio de los tiempos en cada usuario.

La Figura 5.23 muestra una gráfica de tres dimensiones, se diseño así ya que se van a considerar tres aspectos importantes, el usuario, el número de tecla y el tiempo entre cada pulsación. En ésta gráfica se muestran los tiempos de los usuarios que requirieron entre nueve y catorce pulsaciones, es decir, tuvieron un margen de error pequeño, mien-

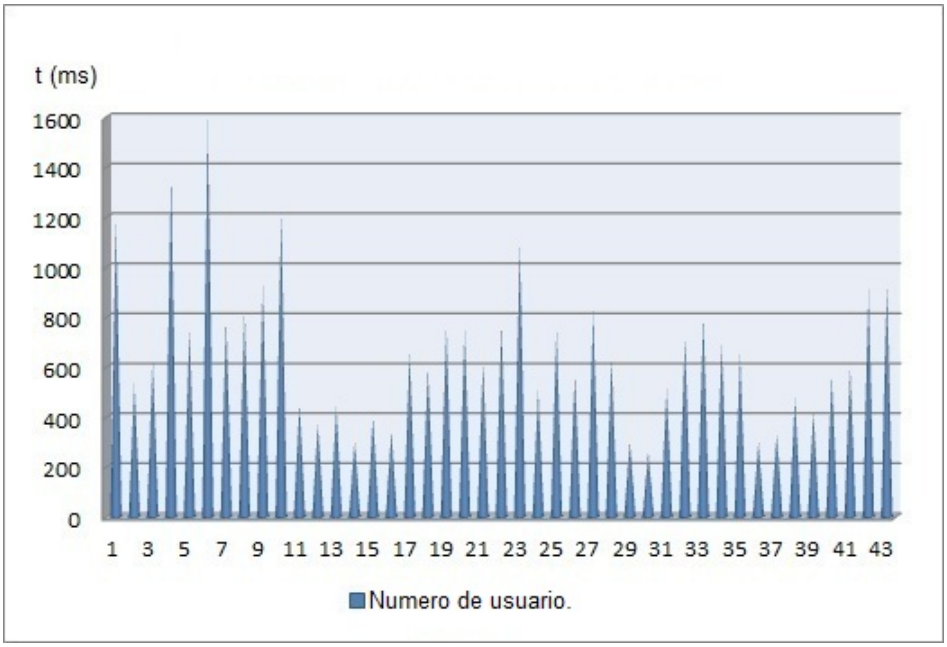


Figura 5.22: Gráfica de la media de los tiempos de cada usuario.

tras que la Figura 5.24 muestra los tiempos para los usuarios totalmente inexpertos ya que requirieron repetir por lo menos tres veces el captcha por colocar letras incorrectas a comparación del captcha original.

La Figura 5.25 es la gráfica general que concentra la información de la muestra que se utilizó para las pruebas en este trabajo. Como puede apreciarse, los tiempos no fueron iguales en ningún caso aunque algunos usuarios realizaron la prueba más de una vez. Lo anterior nos permite concluir que el comportamiento del usuario permite generar información única basada en la primera interacción del usuario con el sistema.

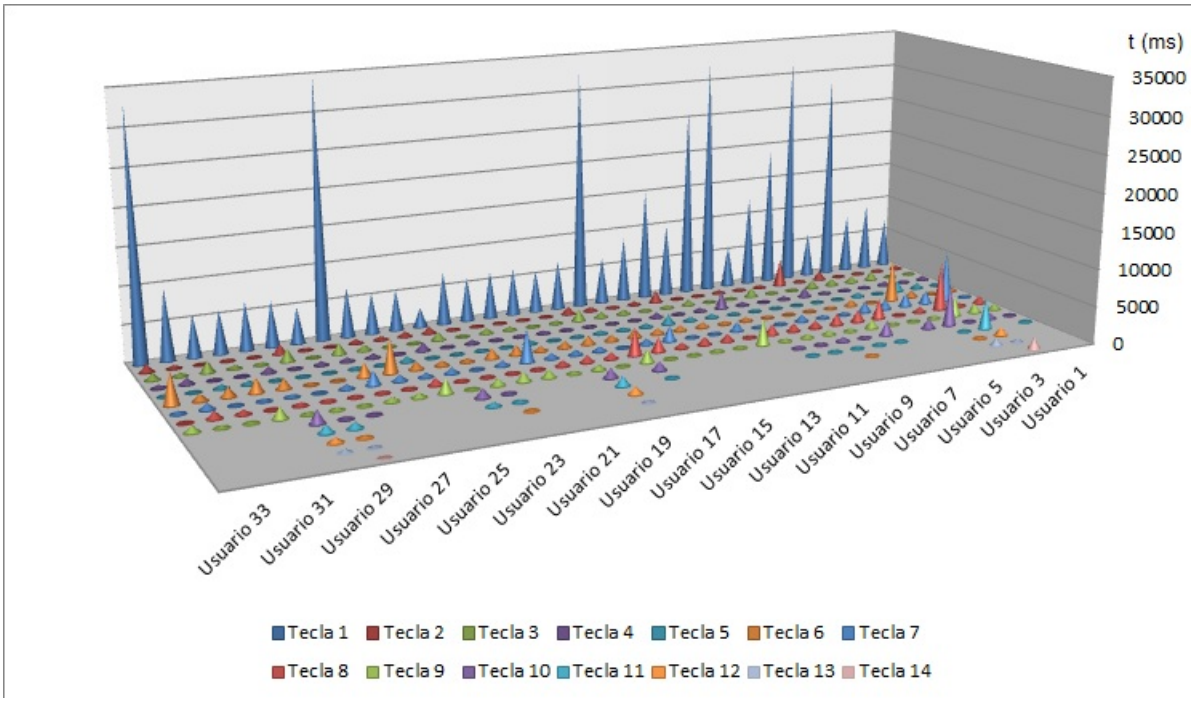


Figura 5.23: Gráfica de los tiempos de entre nueve y catorce pulsaciones .

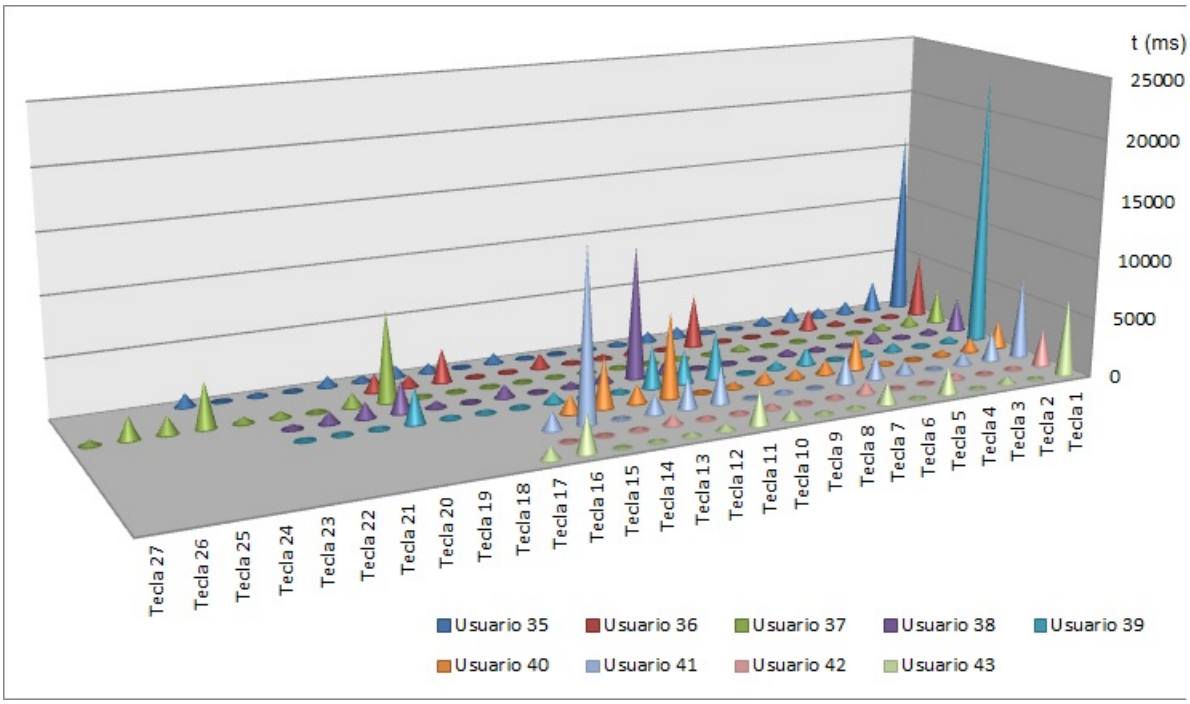


Figura 5.24: Gráfica de los tiempos con mas de nueve pulsaciones .

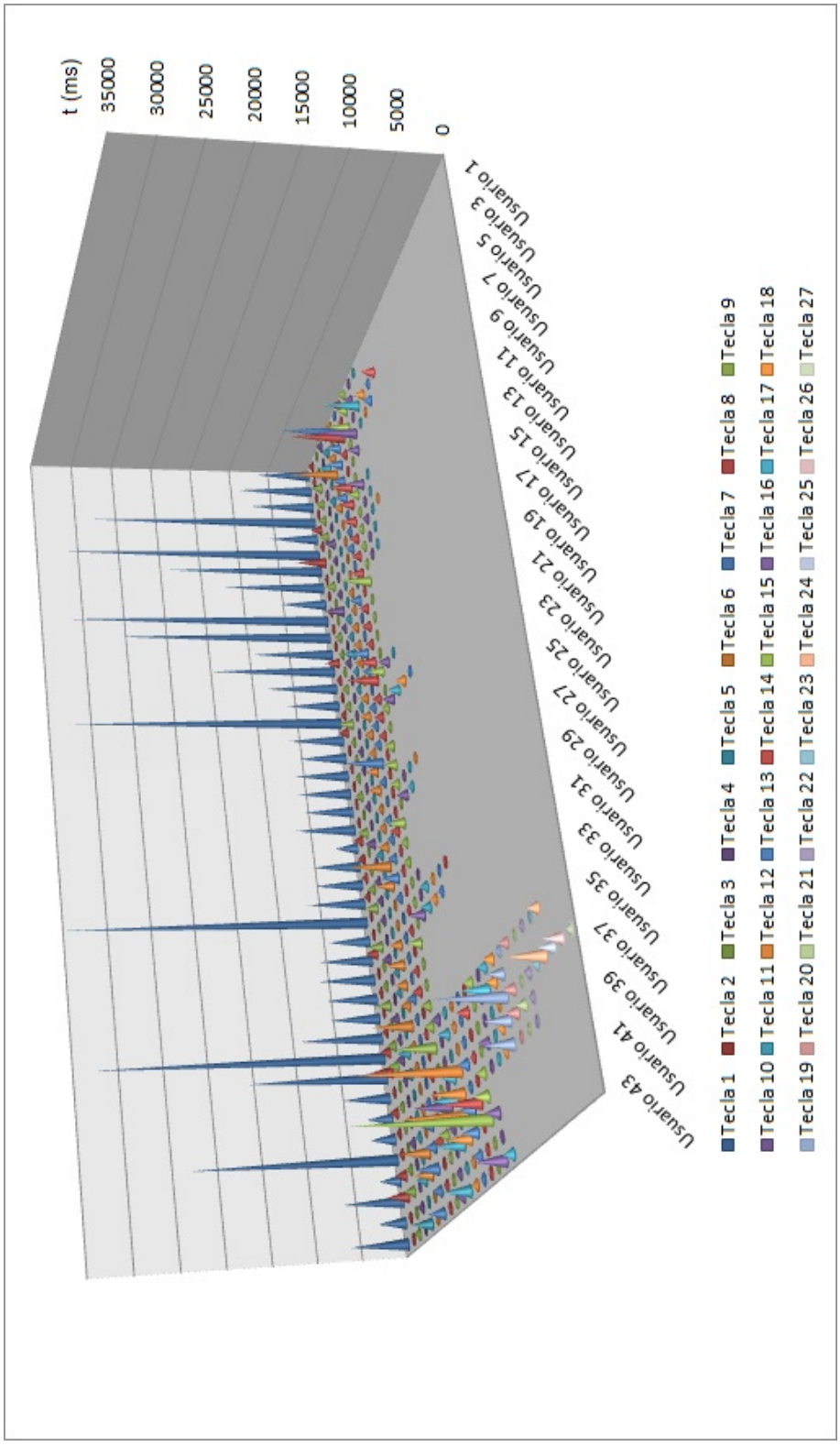


Figura 5.25: Gráfica general.

5.3. Vector de características

Con los tiempos mencionados en la sección anterior, la generación del número semilla es muy simple considerado como un vector de características. Se consideró el formato de las funciones de números pseudoaleatorios, se tomó una función que recibe como argumento una semilla como un arreglo de bytes. Los pasos a seguir para la generación del vector es la siguiente:

1. Se accede al archivo *tiempos.txt* almacenado en el dispositivo móvil.
2. Se toma el arreglo de tiempos que se va creando cada vez que el usuario pulsa una tecla al resolver el captcha.
3. Se convierte el arreglo de tipo entero de longitud grande (*long*) a un arreglo de bytes (vector de características).
4. Se toma el arreglo como argumento a la función pseudoaleatoria.
5. Se genera el número pseudoaleatorio.

El código utilizado para la generación de números semilla se presenta en la Figura 5.26

```
tiemposUsuario.addTimeActivity(System.currentTimeMillis());
NumeroAleatorio folio = new NumeroAleatorio(256, tiemposUsuario.getSeedUser());
NumeroAleatorio factor = new NumeroAleatorio(128, tiemposUsuario.getSeedUser());
```

Figura 5.26: Generación del vector de características.

La Figura 5.27 muestra el código de la generación del vector de características y un ejemplo de ejecución para generación del pseudoaleatorio en NetBeans 7.4.

5.3.1. Aplicación en Firmas digitales

Un ejemplo de la utilización de los números pseudoaleatorios generados a través del vector de características del usuario como número semilla permite que las firmas sean diferentes ya que se usan número pseudoaleatorios diferentes en cada caso.

```
    public static void main(String[] args) {
        Random r = new Random((long)
                573722011821182311359611079652656210231210231D);
        long num = r.nextLong();
        System.out.println(num);
    }
```

```
- Explorer3 (run)
run:
4961115982468162243
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figura 5.27: Generación del número pseudoaleatorio.

Se realizó la programación del algoritmo de firma digital DSA [22] el cual usa dos números pseudoaleatorios que se requiere sean únicos por usuario. Así, primero el usuario resuelve el captcha y después son generados dos números aleatorios que corresponden a la llave privada del usuario y a la llave de sesión para el mensaje firmado.

En los treinta casos de los usuarios elegidos, no se obtuvo ninguna repetición en las firmas. La Tabla 5.1 presenta el vector de características, el número pseudoaleatorio generado y la firma producida, para únicamente 5 usuarios. Como puede observarse, se obtuvo información única por cada intento de solución de los usuarios.

En la tabla 5.2 se puede observar que todos los vectores de caracteristicas son diferentes, se realizaron las pruebas con el sistema funcionando normalmente, por lo tanto los captchas que el usuario resolvio fueron diferentes.

La tabla 5.3 contiene los tiempos de las pruebas realizadas a un segundo usuario, observando dichas pruebas se puede notar que ningúno de los números en esa tabla son iguales, por lo tanto, se cumple que la afirmación mencionada en el inicio del trabajo: Ningún usuario realiza dos cosas de la misma manera.

Tabla 5.1: Generación de firmas basadas en el vector de características.

Usuario	Vector de características Número pseudoaleatorio Firma
1	231811945382153643173119184810075501987247157742521607 165880633836921112635494493245273937825904273180309471 155905608589190718339788776641983537721577597685
2	600813119642116711921352143214139 9 463786688140464707744073547413359 3 597615853508390828976504587418328541039020044922
3	161342653111396478836628887042519713322488712779410475488981943891571123 805429394137062200443357640479504093661749596408780554523976138542725 88 136673316798482995102047727878946488698236832619
4	234945402307335232351500701186402528553615382229908238312346302130224921 7 947939542975475201287134552501551451895527566875847207208583635847192777 859023501172839546689609693923472567891258631902
5	5562586375310806110662397319 35 347609094677605543410074789 55 277704535965511583330642269800126042559914956459

Tabla 5.2: Pruebas realizadas con el mismo usuario y captchas pseudo aleatorios.

Número de solución	Vector de características
1	30734 4402 2771 1735 1199 1882 675 19 2066 362 306 567 4607 2800 580
2	21138 784 17 1898 884 1940 237 4495 7285 2635 1380 2767 4112 1442 750 5 3163 2902 1190 1099
3	7964 995 410 352 750 5 922 1510 1476 481 1986 3165 1238 1731 514 439 346 1020 25 1239 182 178 190 952 447 620 891 48 5765 370 482
4	6966 240 1908 435 958 18 2948 2237 376 511 1692 26 2706 4693 849 1656 185 1320 346 878 6 2158 416 1149 1528
5	10419 630 1745 543 980 13 1905 945 340 2879
6	7121 849 1869 410 921 13 1444 1982 481 591 1963 11 2180
7	7618 485 329 801 1138 67 958 653 869 204 378 1049 35 2604 388 647 611
8	6757 646 1730 669 871 21 1441 484 832 882
9	7472 1292 1919 713 424 383 1375 25 1045 190 213 189 1554 475 312 1134 36 1303 609 706 535
10	7073 650 485 1276 1308 33 1394 821 1060 694

Tabla 5.3: Pruebas realizadas con el mismo usuario y el mismo captcha.

Número de solución	Vector de características
1	5629 879 1649 968 1146 16 2255 2150 605 1740 465 541
2	9776 1426 464 1875 853 18 1595 598 958 894
3	5276 668 974 550 915 13 1726 523 1169 994
4	71191 7975 5349 553 1333 443 6533 15 1513 50 22 1690 570 787 19 4011 3419 2830 1526 2846 16 2224
5	43162 15554 1839 1368 878 3101 11 6573 1630 2033 404 3389 6428 1249 1291 508 342 2823 1565 1658 56 1652 468 1421
6	473 783 2608 1297 218 2940 10555 762 1281 40 1077 523 743 339
7	65296 1333 1705 1814 1139 683 2064 1091 4907 922 1303 282
8	13695 1114 664 848 940 994 1309 1516 624 274
9	6540 1514 2759 556 1563 248 1743 1157 1341 98 3800 600 1118 1087 198 1128 1138 1207 967 824
10	10775 414 761 319 425 10 2438 936 318 211

Capítulo 6

Conclusiones y trabajo futuro

De acuerdo al objetivo general de este trabajo, se desarrolló un sistema para la generación de información única basada en biometría dinámica sobre dispositivos móviles con sistema operativo Android. El sistema denominado SIGENIU utiliza una herramienta común llamada captcha para capturar el comportamiento del usuario a través de obtención de los tiempos requeridos por el usuario al pulsar cada tecla en la resolución del captcha.

Para lo cual, se realizó una investigación sobre las técnicas biometrícas y valorar qué herramienta era la más útil en la obtención del comportamiento del usuario, considerando la tecnología móvil.

La implementación del trabajo se realizó de forma exitosa en las dos infraestructuras planteadas en los objetivos, es decir, sobre dispositivos con sistema operativo Android y sobre computadoras personales con el lenguaje de programación Java.

Al realizar las gráficas del comportamiento del usuario, se pudo analizar la información para comprobar que los números semilla generados por el usuario son únicos, puesto que no existe un límite de valores y mucho menos un rango para restringir al sistema.

La generación de numeros semilla se logró almacenando cada tiempo que al usuario le tomó pulsar cada una de las teclas al resolver el captcha. El arreglo resultante es convertido a un formato en bytes que puede ser correctamente interpretado por la función generadora de números pseudoaleatorios utilizada.

Para demostrar la unicidad de los números semilla, no se realizaron analisis matemáticos, pero basta con observar las gráficas presentadas en la sección anterior, más en especifico la gráfica general, ya que en ella se muestra el total de los vectores de caracteristicas utilizados para este trabajo, y con ello determinar que todas las semillas generadas son unicas.

6.1. Trabajo futuro

Las tareas consideradas como trabajo futuro son las siguientes:

- Modificar el captcha a letras deformadas.
- Crear una interfaz que permita aumentar el tamaño de la gráfica de los tiempos para una visualización clara de los datos.
- Realizar la implementación en dispositivos móviles con otro sistema operativo como Windows Mobile, iOS y BlackBerry.
- Generar el número pseudoaleatorio y utilizarlo en aplicaciones como transacciones electrónicas o autenticación de dos factores.
- Generar en varios formatos el número semilla para que éste pueda ser utilizado en aplicaciones con herramientas más avanzadas como redes neuronales y protocolos de criptografía de llave pública.

Referencias

[1] William Stallings, Lawrie Brown. Hardcover. Computer Security: Principles and Practice (2nd Edition). December, 2011.

[2] A. Bours. Continuous keystroke dynamics: A diferent perspective towards biometric evaluation. Information Security Technology. pp. 36-43, 2012

[3] E. Burnette. Hello, Android: introducing Google's mobile development platform. Pragmatic Bookshelf. 2010

[4] E. Burnette. Hello, Android: introducing Google's mobile development platform. Pragmatic Bookshelf. 2014

[5] E. Maiorana, P. Campisi, N. González-Carballo y A. Neri. Keystroke dynamics authentication for mobile phones. Proceedings of the ACM Symposium on Applied Computing. pp. 21-26. 2011.

[6] R. Moskovich, Identity theft computers and behavioral biometrics. IEEE International Conference on Intelligence and Security Informatics. pp. 155-160. 2009.

[7] C. Fontela. UML, modelado de software para profesionales. Alfaomega. 2011.

[8] M. Fowler y K. Scott. UML, gota a gota. Pearson Addison Wesley. 1999.

[9] M. García-González. Generación de números aleatorios. Reporte técnico, Universidad Autónoma de Barcelona. 2014. *http* : *//www.lawebdefisica.com/apuntsmat/num_aleatorios*.

[10] Garg, R. Rahalkar, S. Upadhyaya y K. Kwiat. Profiling Users in GUI Based Systems for Masquerade Detection. IEEE Information Assurance Workshop. pp. 48-54. 2006.

[11] Google, Inc. ReCaptcha. Google, Inc. 2014. *http* : *//www.google.com/recaptcha/captcha*.

[12] H. Ketabdar, P. Moghadam, B. Naderi y M. Roshandel. Magnetic signatures in air for mobile devices. 14th International Conference on Human-Computer Interaction with Mobile Devices and Services Companion. pp. 185-188. 2012.

[13] Homini. Plataforma Biométrica Homini. Homini S.A. 2004. *http* : *//www.homini.com/new*$_p$*age*$_5$*.htm*.

[14] J. Guerra-Casanova, C. Sánchez-Ávila, G. Bailador y A. Santos-Sierra. Authentication in mobile devices through hand gesture recognition. Journal of Information Security. pp. 65-83. 2012.

[15] José Eduardo Ochoa Jiménez. Función picadillo determinista al grupo G2 y su aplicación en autenticación para dispositivos móviles. Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional. 2013.

[16] K. S. Balgani, V. Phoha, A. Ray y S. Phoha. On the discriminability of keystore feature vectors used in fixed text keystore authentication. Pattern Recognition. pp. 1070-1080.2011.

[17] J. Katz y T. Lindell. Introduction to Modern Cryptography. Chapman & Hall/Crc Cryptography and Network Security Series. 2007.

[18] LaCie. Libro blanco de LaCie: Biometría. LaCie. *http* : *//www.lacie.com/download/whitepaper/WP*$_B$*iometrics*$_E$*S.pdf*.

[19] M. K. Chong, G. Marsden y H. Gellersen. GesturePIN: using discrete gestures for associating mobile devices. 12th International Conference on Human Computer Interaction with Mobile Devices and Services. pp. 261-264. 2010.

[20] M. Karnan, M. Akila y N. Krishnaraj. Review Article: Biometric personal authentication using keystore dynamics. Applied Soft Computing. pp. 1565-1573. 2010.

[21] Open Handset Alliance. Open Handset Alliance. 2009. *http* : *//www.openhandsetalliance.com/index.html*

[22] NIST. Digital Signature Standard (DSS). Federal Information Processing Standards Publications (FIPS PUBS). 1994. *http* : *//www.itl.nist.gov/fipspubs/fip*186.*htm*.

[23] R. Goit, M. El-Abed, B. Hemery y C. Rosengerger. Unconstrained keystore dynamic authentication with shared secret. Computers and Security. pp. 427-445. 2011.

[24] Rodríguez-Henríquez, F. Saqip, N. A. Díaz-Pérez, A. y Koç, C. K. Cryptographic Algorithms on Reconfigurable Hardware (Signals and Communication Technology). Springer-Verlag New York, Inc. 2006.

[25] D. R. Stinson. Cryptography: Theory and Practice. Chapman and Hall/CRC. 2006.

[26] T. Chang, C. T. y J. Lin. A graphical based password keystroke dynamic authentication system for touch screen handheld mobile devices. Journal of System and Software. pp. 1157-1165. 2012.

[27] Carlos Manuel Travieso González, Marcos del Pozo Baños y Jaime Roberto Ticay Rivas. Cuaderno Red de Cátedras Telefónicas. Sistemas Biométricos. p. 21. 2011.

[28] A. Villegas. La tabla de números al azar. Universidad Autónoma de Centro América, Costa Rica. 2012. *http* : *//www.uaca.ac.cr/bv/ebooks/estadistica/*11.*pdf*.

[29] Ciclo de vida de una actividad - Diplomado de Especialización en desarrollo de aplicaciones para Android. Universidad Politécnica de Valencia. 2011. *http* : *//www.androidcurso.com/index.php/tutoriales − android/*37 *− unidad −* 6 *− multimedia − y − ciclo − de − vida/*158 *− ciclo − de − vida − de − una − aplicacion*.

[30] Get the Android SDK - Developers. Google Inc. 2014. *http* : *//developer.android.com/intl/es/sdk/index.html*.

[31] W. N. Bhukya, S. Kumar y A. Negi. Masquerade detection based upon GUI user profiling in linux systems. 12th Asian computing sciencie conference on Advances in Computer Sciencie: Computer and Network Security. pp. 228-239. 2007.

[32] Jairo Gómez. Sistema de huella digital para Bibliotecas. Google Sites. INBIOSYS E.U INGENIERÍA Y SISTEMAS BIOMÉTRICOS. Noviembre, 2012. *https* : *//sites.google.com/site/inbiosys/services/sistema−de−huella−digital−para−bibliotecas*.

[33] M&PFernández. Windows 8 con reconocimiento facial, gracias a Kinect. Windows Noticias. Julio, 2010. *http* : *//www.windowsnoticias.com/windows − 8 − con − reconocimiento − facial − gracias − a − kinect*.

[34] Raúl Sánchez Reíllo. El Iris Ocular como parámetro para la Identificación Biométrica. Escuela Técnica Superior de Ingenieros de Telecomunicación, UNIVERSIDAD POLITÉCNICA DE MADRID. 2000. *http* : *//revistasic.com/revista*41/*agorarevista*$_4$1.*htm*.

[35] Univision.com. Android es el sistema operativo móvil más usado en el mundo. Univision Communications Inc. Noviembre,2013. *http* : *//noticias.univision.com/article/*1738822/2013 − 11 − 13/*tecnologia/noticias/android − es − el − sistema − operativo − movil − mas − usado − en − el − mundo*.

[36] Dra. María de Lourdes López García. Diseño de un protocolo para votaciones electrónicas basado en firmas a ciegas definidas sobre emparejamientos bilineales. Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional CINVESTAV. Unidad Zacatenco. Departamento de Computación.