DISEÑO DE UN SISTEMA DE CONTRASEÑAS GRÁFICO

AUTOR: Sandra Campos Suárez.
RESPONSABLE DE LA ASIGNATURA: Jordi Herrera Joancomartí.
UNIVERSIDAD: Universitat Oberta de Catalunya.
FECHA: Junio 2014.

MÁSTER INTERUNIVERSITARO DE SEGURIDAD DE LAS TECNOLOGÍAS DE LA INFORMACIÓN Y DE LAS COMUNICACIONES.

RESUMEN

El objetivo principal de este proyecto es la creación de un nuevo sistema de contraseña que intente reducir las vulnerabilidades de los sistemas actuales, como el sistema de contraseña gráfica de Google Android. La principal vulnerabilidad de estos sistemas consiste en analizar los restos de residuos de grasa que los dedos dejan en la pantalla (screen scrapers).

Para realizar este diseño se pensó en los dispositivos móviles actuales y en concreto en el sistema de contraseña que incorpora Android. Con ese sistema se acumulan residuos siempre en el mismo sitio, lo cual lo hace bastante vulnerable a los ataques anteriores. Con esto se piensa en hacer un sistema en el que cuando se introduzca el patrón para desbloquear la pantalla no se dejen residuos siempre en el mismo sitio, que sea aleatorio cada vez. Otra de las cosas que se tuvo en cuenta a la hora de pensar en este nuevo diseño, fue otra vulnerabilidad conocida como "shoulder surfing", un ataque más bien de ingeniería social ya que consiste en averiguar la contraseña o patrón mirando la pantalla por detrás de la persona, sin que se dé cuenta. Con esto se pensó en que cuando se introduce el patrón, no se vea lo que se está pulsando.

Por otro lado, se pensó en algo que fuera como un juego y que atraiga a un público más o menos antes. La primera idea fue hacer algo parecido al mítico juego de "simon dice". Con esto se llegó a la conclusión de realizar el sistema pensando en patrones de colores.

Todo diseño requiere de una implementación posterior. Se decidió por los dispositivos Android ya que es más fácil para programar y probar porque una gran mayoría de personas posee esto sistemas. Se ha realizado un prototipo en esta plataforma para probarlo en algunos voluntarios y así evaluar el producto resultante. Después de realizar las pruebas podemos concluir en que los usuarios les parecía divertido y diferente, y la mayoría no tenían problemas a la hora de recordar la contraseña. También que suelen elegir los mismos colores para poder recordarlos mejor.

Para concluir, se cree que el sistema propuesto reduciría las vulnerabilidades

ÍNDICE

Introducción

1. Sistemas de contraseña gráficas.
- 1.1 Recall-based Systems.
- 1.2 Recognition-based Systems.
- 1.3 Cued-call Systems.
- 1.4 Sistema elegido para el diseño.

2. Descripción y análisis del nuevo sistema.
- 2.1 Objetivo del sistema.
- 2.2 Análisis del sistema.

3. Diseño del sistema propuesto.
- 3.1 Tipo de aplicación y planificación.
- 3.2 Funcionamiento.

4. Implementación.
- 4.1 Interfaces del usuario.
- 4.2 Descripción del código.

5. Análisis de seguridad del sistema propuesto.
- 5.1 Espacio de contraseñas.
- 5.2 Encriptación - Codificación.
- 5.3 Vulnerabilidades.

6. Pruebas del sistema.
- 6.1 Interacción con los voluntarios.
- 6.2 Resultados y conclusiones.

Bibliografía.

Introducción

El objetivo del proyecto es realizar un sistema de contraseñas gráfico, el cual implica que tiene que tener algún componente gráfico a la hora de autenticarse y no caracteres alfanuméricos.

Los sistemas de autenticación convencionales, basados en usuario y contraseña, presentan graves problemas ya que una contraseña segura es difícil de recordar, lo que lleva a los usuarios a poner contraseñas fáciles que son muy vulnerables a ataques simples de diccionario o con ingeniería social. A causa de estos problemas se empiezan a utilizar sistemas de contraseñas basados en gráficos, como imágenes, formas, colores que según estudios científicos son más fáciles de recordar que los clásicos sistemas de autenticación.

Sistemas de contraseña gráfica

Para poder diseñar este nuevo sistema ha sido necesario conocer los tipos de sistemas de contraseña que hay en la actualidad para así realizar una mejor elección.

A continuación se explica brevemente la clasificación de estos sistemas y una gran variedad de los que existen.

Recall-Based Systems

En estos sistemas, los usuarios habitualmente escriben sus contraseñas en un lienzo en blanco o en una cuadricula.

Draw-a-secret (DAS) [Jermyn et al. 1999]: fue el primer sistema de contraseña gráfica propuesto. Los usuarios dibujaban su contraseña en un grid 2D utilizando un stylus o un ratón. El sistema codificaba la contraseña dibujada como coordenadas en el grid según el trazado. Utilizando este sistema aparece BDAS [Dunphy and Yan 2007], que es igual pero añadiendo una imagen de fondo para que los usuarios se animase a crear contraseñas más complejas.

Passdoddle [Goldberg et al. 2002, Varenhorst 2004]: Es parecido al DAS pero los usuarios dibujan la contraseña sin un grid visible. Se le añaden distintos grosores del trazo y colores.

PassShapes [Weiss and De Luca 2008]: las contraseñas son traducidas a caracteres alfanuméricos mediante 8 direcciones distintas del trazo a intervalos de 45°. La contraseña no hace falta que se dibuje en la misma posición, puede dibujarse en cualquier parte de la cuadrícula.

Pass-Go Scheme [Tao and Adams 2008]: los usuarios dibujan sus contraseñas utilizando los puntos de intersección en el grid. Se capturan los movimientos de los trazos. El espacio de contraseñas es mayor que en DAS debido a que se pueden realizar movimientos diagonales. El sistema de Google Android es una implementación de este tipo.

GrIDSure [2009]: es un producto comercial que muestra dígitos en una cuadrícula de 5x5. Los usuarios seleccionan y memorizan un patrón ordenado dentro de los 25 cuadros que componen la cuadrícula y entran los dígitos utilizando el teclado. En las autenticaciones los dígitos se muestran colocados aleatoriamente.

Recognition-based Systems

En este tipo de sistemas, los usuarios memorizan un conjunto de imágenes durante la creación de la contraseña. En las autenticaciones tienen que reconocer esas imágenes dentro de otra secuencia. Se basa en la capacidad que tiene el ser humano en reconocer imágenes que ha visto previamente.

PassFaces [Passfaces Corporation 2009]: los usuarios seleccionan un conjunto de caras humanas. Durante la autenticación se presentan unas caras candidatas. El usuario debe seleccionar la cara que pertenece a su conjunto. Esto se repite varias veces antes de completar el proceso de autenticación.

Story [Davis et al. 2004]: los usuarios seleccionan una secuencia de imágenes en un portfolio. Para autenticarse, los usuarios deben seleccionar su secuencia en el orden correcto.

Dejà Vu [Dhamija and Perrig 2000]: los usuarios seleccionan y memorizan un conjunto aleatorio de imágenes de arte. Después tienen que reconocer las imágenes dentro de un conjunto mayor.

Cued-call Systems

Estos sistemas consisten en que los usuarios seleccionan y memorizan posiciones o zonas dentro de una imagen, en identificar ubicaciones específicas.

PassPoints [Wiedenbeck et al. 2005a, 2005b, 2005c]: la contraseña consiste en una secuencia de 5 puntos o click-points dentro de una imagen dada por el sistema. En la fase de autenticación, los usuarios deben reproducir estos puntos en el orden y situación correctos.

Cued Click-Points (CCP) [Chiasson et al. 2007b]: los usuarios seleccionan una zona con un click en cada una de las 5 imágenes que se presentan en secuencia. Cada imagen se almacena junto con las coordenadas donde se hizo click, Una variante de este sistema es PCCP [Chiasson et al. 2008a], en la que se intenta persuadir al usuario para que seleccione puntos más aleatorios.

Sistema elegido para el diseño

Después de comprender cada uno de los distintos sistemas, se ha optado por la opción de realizar nuestro sistema siguiendo

los principios de un sistema Recall-Based. Se ha elegido este tipo porque se adapta mejor con el tipo de diseño que se va a emplear, en el cual se utiliza una cuadrícula para introducir la contraseña. Es parecido al sistema de PassShapes porque los valores de los colores introducidos son traducidos a caracteres alfanuméricos.

Uno de los sistemas más famosos y conocidos es el que utiliza Google Android en sus dispositivos móviles. Debido a tan auge, este sistema se ha visto comprometido debido a la multitud de ataques y vulnerabilidades que tiene. La más importante es la que analiza los residuos que dejan los dedos en la pantalla para así poder averiguar el patrón introducido por el usuario.

Por eso se ha decidido este sistema, mayormente por mejorar este tipo de vulnerabilidad.

Descripción y análisis del nuevo sistema

Objetivo del sistema

El objetivo del sistema es evitar la mayoría de ataques que se producen analizando los residuos que se quedan en las pantallas de los dispositivos móviles. Es sorprendente lo difícil que es hacer desaparecer estos residuos y lo fácil que es analizarlos para obtener el patrón que se haya introducido.

Por esta razón se intenta crear un sistema en el que aún analizando estos residuos no se pueda adivinar el patrón que se ha introducido, y por tanto la seguridad será mucho mayor.

La clave del sistema diseñado para conseguir esta meta, es que a la hora de introducir el patrón, las opciones donde pulsar el usuario aparezcan de manera aleatoria, así que aunque el patrón sea siempre el mismo, los residuos que se dejan en la pantalla no estarán situados en la misma zona, lo que sería muy fácil de analizar, sino que estarán por toda la pantalla y será más complicado de analizar por parte de alguien que quiera obtenerlo.

A parte de esto, el sistema ofrece una amplia combinación de patrones, ya que el usuario puede elegir entre 20 colores para formar su patrón, y obligatoriamente tendrá que introducir entre 4 y 8 colores. Para facilitar la memorización, los colores que se elijan podrán repetirse, con lo que se tienen más combinaciones.

También se añade un control para evitar que cualquier persona que mire por detrás le sea más difícil adivinar el patrón. Este control es que no se iluminen los colores que se pulsan y además de que cada 5 segundos los colores cambian de posición.

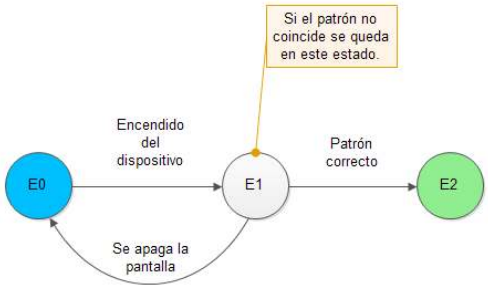
Análisis del sistema

Para analizar el nuevo sistema se han realizado una serie de diagramas y máquinas de estados para representar todas las acciones que se pueden realizar.

Hay que aclarar que para realizar el sistema se ha hecho una implementación en la que aparece un menú para elegir las opciones. Evidentemente si el nuevo diseño estuviese integrado en el sistema operativo no aparecería este menú sino que a la hora de desbloquear el dispositivo ya saldría para poner el

patrón (previamente se habría introducido en las opciones del dispositivo nuestro patrón).

Estado inicial:

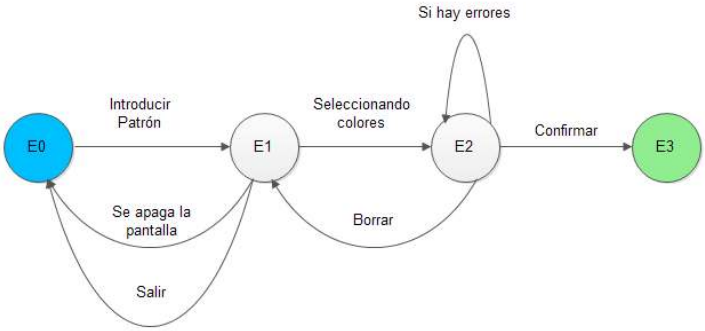


E0 indica el estado en reposo, es decir, el estado de la aplicación antes de iniciarse. Si encendemos la pantalla del dispositivo pasaríamos al estado E1.

En el estado E1 tendríamos la cuadrícula de 20 colores (5x4) para introducir el patrón. Estaríamos siempre en este estado si no se introduce el patrón correcto. Cuando se introduce el patrón correcto pasaríamos al estado E2. Si apagamos la pantalla volvemos al estado E0.

El estado E2 indica que la aplicación terminó correctamente, lo cual implica que el patrón introducido era correcto y se ha desbloqueado el dispositivo.

Máquina de estados para Introducir el patrón:



El estado E0 es el estado inicial donde aparece el menú. Si pulsamos en introducir patrón pasamos al estado E1 que es la pantalla principal para empezar a poner el patrón.

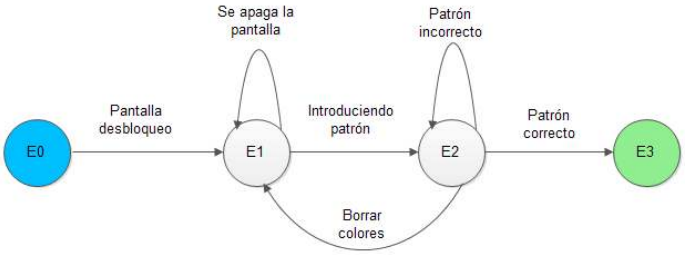
Cuando empecemos a introducir colores pasaremos al estado E2 en el que ya hay colores elegidos. Si se apaga la pantalla o le damos a salir, volvemos al estado E0.

El estado E2 es el estado en el que se encuentra el sistema cuando hemos estado pulsando botones. Nos quedaremos en este estado si hay errores, por ejemplo que al confirmar el número de elementos sea menor que 4, o superemos los 8 colores. Si le damos al botón Borrar regresaremos al estado E1 en el que no se ha pulsado ningún color.

El estado E3 es el estado final, lo cual quiere decir que hemos confirmado un patrón correcto.

El diagrama de estados de modificar el patrón es similar a este.

Máquina de estados de Desbloquear pantalla:



El estado E0 es el estado de reposo antes de iniciar el proceso de desbloqueo. Si pulsamos en el desbloqueo del móvil pasaríamos al estado E1 (en el prototipo sería cuando pulsásemos en el botón de desbloquear pantalla).

El estado E1 es el estado inicial cuando aparecen los colores aleatorios en la pantalla. Cuando se empiezan a pulsar los colores pasaríamos al estado E2. Si bloqueamos la pantalla, al encenderla seguiríamos en el estado E1.

El estado E2 indica el proceso en el cual se van pulsando los colores para introducir el patrón. Siempre que el patrón sea incorrecto permaneceríamos en este estado. Si borramos lo que estamos introduciendo pasaríamos al estado E1. Si el patrón es correcto avanzamos hacia el estado E3.

E3 es el estado en el que la aplicación terminó correctamente, lo que quiere decir que el patrón introducido es correcto y se ha desbloqueado el dispositivo.

Diseño del sistema propuesto

Para realizar un buen diseño del sistema tenemos que tener claro qué tipo de aplicación queremos realizar, en qué plataforma o para qué público está orientado. Teniendo claro estos pasos previos podemos realizar un diseño correcto.

Tipo de aplicación y planificación

La aplicación ha sido diseñada para ser ejecutada en la plataforma Android. No se pretende sustituir el sistema actual, sino realizar un prototipo y simulación de lo que sería éste. Se utilizará un lenguaje de programación orientado a objetos, java, junto con el SDK de Android.

La aplicación está dividida en varias funciones:

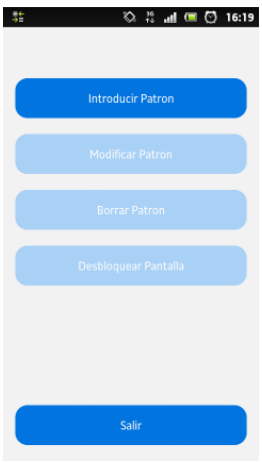
- Introducir patrón.
- Modificar patrón.
- Borrar patrón.
- Desbloquear pantalla.

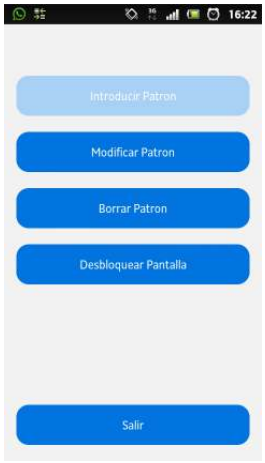
La paleta de colores consiste en 20 botones cuadrados cada uno representando un color diferente. En las funciones de introducir y modificar patrón aparecerá otra paleta de máximo 8 botones que se rellena con los colores que vamos pulsando.

Se capturan todos los eventos de las acciones de pulsar colores o botones y se procesan adecuadamente según la situación. Todos los diferentes estados de la aplicación son controlados(borrar patrón, confirmar,...).

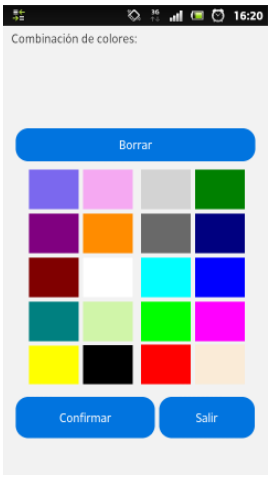
Funcionamiento

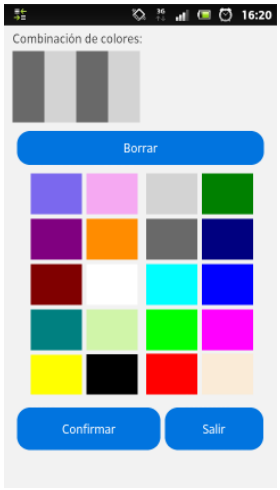
Para poder simular el comportamiento del sistema, al comienzo se visualizan las opciones que se pueden realizar. Al principio sólo está activa la opción de introducir patrón (aparte de la de Salir). Esto es así porque se necesita tener registrado el patrón previamente para poder desbloquear la pantalla. Cuando ya se ha registrado el patrón esa opción se desactiva y se activan las demás.





En el proceso de introducir patrón y modificar patrón nos encontraremos con la paleta de colores que se seleccionan pulsando con los dedos y se irán añadiendo en la paleta de 8 colores de la parte superior.





En cualquier momento podemos borrar la combinación que vamos seleccionando, por si queremos empezar de nuevo. Con el botón Confirmar se guardará el patrón elegido.

Después de seleccionar el patrón y guardar los cambios la aplicación está lista para desbloquear la pantalla.

En la pantalla de desbloqueo, ahora en el prototipo se ha añadido un botón para borrar lo que estás introduciendo, ya que al cambiar los botones de colores cada 5 segundos puede que se pulse un color erróneo, y con esta opción podemos resetear lo que hayamos pulsado.

Al pulsar en validar nos dirá si la combinación es correcta o no. Si no es correcta no saldrá de esta pantalla.

El interface se ha intentado hacer lo más fácil y amigable posible, para ayudar al usuario a realizar las opciones.

Implementación

La implementación como se ha dicho anteriormente se ha hecho con el SDK de Android. Concretamente se ha usado como mínimo la API 9, ya que el dispositivo móvil que se usaba para las pruebas tiene la versión de Android 2.3.7. Pero sirve para versiones superiores.

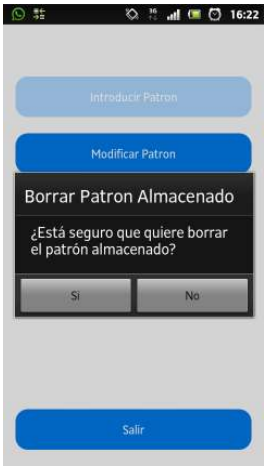
Interfaces de usuario.

En el diseño del sistema ya se ha realizado una introducción los interfaces del usuario. Aquí lo explicaremos más detalladamente.

En el proyecto existen cuatro actividades principales. La primera de ellas es la actividad que muestra el menú. La segunda y tercera son para Introducir Patrón y Modificar Patrón respectivamente. La cuarta y última actividad es la de Desbloquear Pantalla. Además de estas cuatro actividades existe una clase de gestión, que es la que guarda el patrón, los códigos para encriptar y los códigos que se asignan a los botones para poder cargar los colores. En la actividad principal se crea un objeto de la clase y este se pasa al resto de actividades (para poder hacer esto, la clase de gestión tienen que implementar la clase Parcelable). He optado por implementar esta forma en vez de guardarlo en un archivo porque pienso que es más segura porque si se guardase en un fichero o en la memoria del teléfono, podría estar sujeta a otros ataques para recuperar esa información. De todas formas hay que aclarar que es un prototipo, y como todo prototipo se pueden realizar cambios para mejorar el sistema.

En las actividades de Introducir patrón y modificar patrón se controla que el patrón tenga cuatro colores o más cuando se le da al botón confirmar. Si seguimos pulsando colores hasta el final, cuando se superen los ocho, nos sale el mensaje de advertencia, ya que no se puede introducir más de 8. Por otro lado tenemos el botón borrar, que nos deja resetear los colores que pulsemos para empezar de nuevo.

En el menú principal también disponemos del botón borrar, el cual nos permite borrar el patrón que está almacenado y se desactivarían los botones menos el de Introducir Patrón que volvería a estar activo. Es como si se empezara de nuevo. Con el botón salir, salimos de la aplicación.



La última actividad, que es la de Desbloquear Pantalla, no presenta ninguna complicación. Los colores aparecen colocados aleatoriamente cada vez que se inicia este evento, y cada 5 segundo se van cambiando. Anteriormente se ha explicado el botón borrar, que es para borrar lo que llevamos pulsado, y el botón confirmar que comprueba si la combinación introducida coincide con el patrón guardado.

Descripción del código

En este apartado vamos a describir las partes más importantes ya que el código es bastante amplio.

Para tener todo encapsulado, se ha creado una clase para gestionar y almacenar todos los valores que se necesitan por

igual en casi todas las actividades. Esta clase es la clase Tgestion.

En esta clase se almacenan los identificadores de los botones de los 20 colores y de los 8 colores elegidos para guardar. Este identificador sirve después para asignarlos como id en las actividades de introducir y modificar patrón.

```
private void inicializarBotones(){
		//---- Botones para los colores disponibles
		String nombre_colores="buttonCol";
		String nombre_comb="buttonComb";

		for(int i=0;i<this.botones_colores.length;i++)
		{

			this.botones_colores[i]=nombre_colores+i;
		}

		for(int i=0;i<this.botones_combinacion.length;i++)
		{

			this.botones_combinacion[i]=nombre_comb+i;
		}
	}
```

Esta clase también incluye el string con la contraseña encriptada en md5. Para que se entienda, cada color tiene asignado un valor (un conjunto de caracteres), conforme se van pulsando los colores se van concatenando estos caracteres y el resultado final se encripta en md5. En otro apartado se verá el procedimiento.

Lo más importante de esta clase es que tiene que implementar la clase Parcelable para así poder pasar por parámetro el objeto entre actividades. Se tienen que sobrescribir los métodos principales:

```
@Override
	public int describeContents() {
		// TODO Auto-generated method stub
		return 0;
	}

	@Override
	public void writeToParcel(Parcel destino, int flags) {
		// TODO Auto-generated method stub
		destino.writeStringArray(this.botones_colores);
		destino.writeStringArray(this.botones_combinacion);
		destino.writeStringArray(this.secret);
		destino.writeStringArray(this.cod_colores);
	}

	private void readFromParcel(Parcel in){
		in.readStringArray(this.botones_colores);
		in.readStringArray(this.botones_combinacion);
		in.readStringArray(this.secret);
		in.readStringArray(this.cod_colores);
	}
```

```
        public static final Parcelable.Creator<Tgestion> CREATOR = new
Parcelable.Creator<Tgestion>() {

            @Override
            public Tgestion createFromParcel(Parcel source) {
                return new Tgestion(source);
            }

            @Override
            public Tgestion[] newArray(int size) {
                return new Tgestion[size];
            }
        };
```

Para saber que botón hemos pulsado y que clave le corresponde, tenemos una variable hashtable en la que se guarda la clave-valor. La clave es el id del botón que se pulsa y el valor son los caracteres de ese color. En una función se recorre esta tabla para comprobar que botón es pulsado y se van guardando los caracteres concatenados.

```
        private void asignarClaves(){
                this.claves_colores.put(this.botones_colores[0].getId(), "#008080_Gre000");
                this.claves_colores.put(this.botones_colores[1].getId(), "#0000FF_Blu000");
                this.claves_colores.put(this.botones_colores[2].getId(), "#FF00FF_Mag255");
                this.claves_colores.put(this.botones_colores[3].getId(), "#FF0000_Red255");
                this.claves_colores.put(this.botones_colores[4].getId(), "#FFFF00_Yel255");
                this.claves_colores.put(this.botones_colores[5].getId(), "#00FF00_Gre000");
                this.claves_colores.put(this.botones_colores[6].getId(), "#00FFFF_Cya000");
                this.claves_colores.put(this.botones_colores[7].getId(), "#FFFFFF_Whi255");
                this.claves_colores.put(this.botones_colores[8].getId(), "#696969_Gra105");
                this.claves_colores.put(this.botones_colores[9].getId(), "#008000_Gre000");
                this.claves_colores.put(this.botones_colores[10].getId(), "#FF8C00_Ora255");
                this.claves_colores.put(this.botones_colores[11].getId(), "#FAEBD7_Whi250");
                this.claves_colores.put(this.botones_colores[12].getId(), "#800080_Pur128");

                this.claves_colores.put(this.botones_colores[13].getId(), "#F5A9F2_Pin245");
                this.claves_colores.put(this.botones_colores[14].getId(), "#7B68EE_Blu123");
                this.claves_colores.put(this.botones_colores[15].getId(), "#D3D3D3_Gra211");
                this.claves_colores.put(this.botones_colores[16].getId(), "#000080_Blu000");
                this.claves_colores.put(this.botones_colores[17].getId(), "#D0F5A9_Gre208");
                this.claves_colores.put(this.botones_colores[18].getId(), "#000000_Bla000");
                this.claves_colores.put(this.botones_colores[19].getId(), "#800000_Red128");
        }

public void addColorToComb(Integer id){
                String v;
                Integer k;

                Enumeration<String> value = this.claves_colores.elements();
                Enumeration<Integer> keys = this.claves_colores.keys();

                while(keys.hasMoreElements() && value.hasMoreElements()){

                        k=keys.nextElement();
                        v=value.nextElement();
                        if(k.equals(id)){
                                this.comb_comprobar[this.valoresColores8]=v;
                                this.comb_desc=this.comb_desc+v;
                        }
                }
        }
```

Cuando confirmamos el patrón que queremos como contraseña, se encripta con md5 la cadena entera que contiene los valores de los colores concatenados.

En la actividad de desbloquear la pantalla los colores están guardados en una lista (exactamente el color html para después asignar el fondo del botón). La lista se lee para asignar los colores de manera aleatoria. En esta actividad tenemos dos hastables, uno que guarda el id con el color html, y otro el color html con la cadena asignada a ese color. Cuando se pulsa en un color se recorren estas dos variables para obtener al final la cadena, la cual se va concatenando con los otros colores que se pulsan.

En el momento en el que se dé al botón desbloquear, se encriptará esta cadena y se comprueba con el valor encriptado del patrón introducido como contraseña que se almacena en el objeto de la clase Tgestion:

```
public void ComprobarPatron(View v){
		String [] combinacion_guardada=new String[1];
		String combinacion_enc=new String();

		if( (this.comb_desc!=null) && (!this.comb_desc.equals("")) ){
			combinacion_guardada=this.gestion.getSecret();

			try {
				combinacion_enc=getMD5(this.comb_desc);

			} catch (Exception e) {
				e.printStackTrace();
			}

			if(combinacion_guardada[0].equals(combinacion_enc))
			ventanaAcierto();
			 else
			ventanaError();
		}
		else
			this.informacionValidar();

	}
```

Análisis de seguridad del sistema propuesto

En este apartado se describen los métodos usados para controlar las vulnerabilidades citadas en el objetivo del diseño y un análisis de éstas.

Espacio de contraseñas

En un principio se pensó en la idea de no repetir colores a la hora de elegir el patrón. Pero al final se llegó a la conclusión, y de hecho se implementa, de que si se puedan repetir colores. Esto significa que aumentaríamos el número de combinaciones posibles y a la vez lo ponemos más fácil a los usuarios para que puedan recordar mejor los patrones.

Ya que se pueden repetir los colores, se dispone del número máximo de combinaciones. Como la aplicación exige como mínimo 4 colores, disponemos de un total de 20^4 combinaciones posibles. En cambio si se selecciona un patrón con 8 colores, que es lo máximo permitido, tendríamos un total de 20^8 combinaciones, que son muchas. A continuación está la tabla:

- 4 colores --> 20^4= 160.000
- 5 colores --> 20^5= 3.200.000
- 6 colores --> 20^6= 64.000.000
- 7 colores --> 20^7= 1.280.000.000
- 8 colores --> 20^8= 25.600.000.000

En total serian: 26.947.360.000 combinaciones permitidas, que equivale al espacio total de posibles claves. Con esto podemos ver que son muchas combinaciones por lo que resulta más seguro que otros sistemas actuales.

Encriptación - Codificación

El algoritmo elegido para codificar el patrón es el md5.

Cada color o botón tiene asociado un código o cadena único. Cuando introducimos el patrón, cada código de color se va concatenando para juntar una única cadena. La cadena final se codifica con md5 y se guarda para posteriormente compararlo en la fase de desbloqueo.

Cuando estemos introduciendo el patrón para desbloquear la pantalla se hace como lo anterior, se codifica en md5 y se

compara con la cadena guardada. Si los dos coinciden significa que es el mismo patrón, por tanto se desbloquea la pantalla.

Vulnerabilidades

Tal y como está diseñado el sistema, si una persona nos mira el dispositivo sin que nos demos cuenta cuando introducimos el patrón, resulta bastante difícil adivinar lo que se está poniendo. Esto se consigue gracias a que cuando se pulsan los colores no tienen ningún efecto de iluminación por ejemplo, pero sobretodo se debe a que los colores van cambiando cada 5 segundos de posición.

En lo referente al screen scraper con la localización aleatoria de los colores logramos que los residuos en la pantalla no estén siempre en el mismo sitio por lo que resulta mucho más difícil sacar el patrón con esta técnica.

Pruebas del sistema

El sistema ha sido probado en un móvil Xperia U, que se le ha facilitado a los voluntarios. Algunos de ellos se han instalado el .apk en sus teléfonos.

Interacción con los voluntarios

Se han realizado pruebas de la aplicación con voluntarios, más o menos de distintas edades, para que opinen y testen el producto. No se les han dado pautas ni sugerencias de cómo poner los patrones.

La mayoría de las personas que han probado la aplicación suelen poner únicamente 4 colores y repetidos (alrededor del 80%), porque lo recuerdan mejor a la hora de desbloquearlos. Cuando utilizan un patrón más largo, a veces tienen problemas en recordarlo, sobretodo pasado un tiempo.

En general ha habido de todo tipo de opiniones. Algunos voluntarios les parece interesante que se base en colores e intuitivo, y otros opinan que a veces es difícil de recordar. También han tenido problemas con la recolocación de los colores cada 5 segundos, porque al cambiar justo cuando pulsaban no le daban al color correcto y tenían que volver a empezar.

Resultados y conclusiones

En general, el primer prototipo entre los usuarios ha tenido buena aceptación. Se pueden realizar algunas mejoras para que el sistema sea más efectivo.

También en lo referente a la programación, se pueden cambiar varios aspectos para que la aplicación fluya más sin parones o sea más rápida a la hora de comprobar el patrón. También hay que citar que no se han diseñado plantillas para todos los tamaños de los distintos dispositivos, por lo que con este prototipo en algunos terminales no se verá adecuadamente.

Ha habido varios retoques en la implementación con respecto al diseño previo. Por ejemplo uno sería el de añadir el botón borrar mientras se pide el desbloqueo de la pantalla.

Para concluir, el sistema en general es bastante seguro, si se utilizase con ciertas recomendaciones para elegir el patrón lo sería más.

Bibliografía

XIAOYUAN SUO, YING ZHU, G.SCOTT.OWEN. Graphical Passwords: A Survey.

ARASH HABIBI LASHKARI, SAMANEH FARMAND, DR. OMAR BIN ZAKARIA, DR. ROSLI SALEH. 2009, vol 6, N°3. Shoulder Surfing attack in graphical password authentication.

ARASH HABIBI LASHKARI, SAMANEH FARMAND, DR. OMAR BIN ZAKARIA, DR. ROSLI SALEH. 2009, vol 6, N°3. A wide-range survey on Recall-Based Graphical User Authentications algorithms based on ISO and Attack Patterns.

SHUSHUANG MAN, DAWEI HONG, MANTON MATTHEWS. A Shoulder-Surfing Resistant Graphical Password Scheme - WIW.

ROBERT BIDDLE, SONIA CHIASSON, P.C. VAN OORSCHOT. Graphical Passwords: Learning from the First Twelve Years.

IAN JERMYN, ALAIN MAYER, FABIAN MONROSE, MICHAEL K. REITER, AVIEL D. RUBIN. 1999. The design and analysis of graphical passwords.

HAI TAO, CARLISLE ADAMS. 2007. Pass-Go: A proposal to improve the usability of graphical passwords.

O.O AYANNUGA PHD. 2012. A review of the security and usability features of different graphical password authentication schemes.