Mejoras de un sistema de contraseñas graficas

Autor: Jose Luis Brehcist Rodríguez
Responsable de la asignatura: Jordi Herrera Joancomartí
Universidad: Universitat Autònoma de Barcelona
Fecha: 03-01-2014

Máster interuniversitario de Seguridad de las tecnologías de la información y de las comunicaciones

UOC Universitat Oberta de Catalunya www.uoc.edu

UAB Universitat Autònoma de Barcelona

Universitat Rovira i Virgili

UIB Universitat de les Illes Balears

Resumen

Este trabajo se realiza con el objetivo de mejorar la seguridad del sistema de contraseñas grafico del sistema operativo de Google Android, el cual se ha convertido en uno de los sistemas de contraseñas gráficos más usados en la actualidad. Esta popularidad lo ha convertido también en uno de los sistemas más atacados en busca de vulnerabilidades. Una de las vulnerabilidades encontradas, es la de analizar los residuos de grasa que los dedos dejan en la pantalla tactil, para averiguar el patrón de desbloqueo utilizado. Este ataque es sorprendentemente sencillo de realizar y alarmantemente efectivo.

Para diseñar la propuesta del sistema nuevo nos basamos en el uso diario de los teléfonos inteligentes, donde se utiliza el dedo como dispositivo de selección y desplazamiento, entre otras cosas. Estos movimientos normalmente se producen en el centro de la pantalla y debido a esta característica, se propone aumentar el número de puntos de la cuadricula. Dejando ahora en el centro de la pantalla cuatro puntos en donde antes solo había uno. Con esta idea, se espera dificultar la obtención del patrón usado por el usuario, ya que si se usan los puntos del centro, los residuos de grasa dejados al desbloquear la pantalla se podrían ver alterados con los residuos dejados con el uso diario del dispositivo.

A la idea de aumentar los puntos de la cuadricula, se le añade la selección de un color. De esta manera si el usuario decide elegir una secuencia de puntos sencilla y que no utilice los puntos del centro, aun así, se habrá conseguido aumentar el nivel de seguridad. Ya que anteriormente solo había que averiguar la dirección del patrón y ahora hay que multiplicar esas dos opciones por los nueve posibles colores que pueda tener.

Para evaluar la aceptación del sistema propuesto, se ha realizado un prototipo para la plataforma Android y se ha buscado a un grupo de voluntarios muy diferenciados para probarlo. Con este proceso, se intenta lograr una muestra representativa de usuarios potenciales para este sistema si se comercializase.

Una vez analizadas las pruebas, podemos concluir que los usuarios no han tenido problemas para recordar la secuencia elegida en el sistema propuesto, aunque es algo más complejo que el sistema actual. Con estas pruebas también se ha podido demostrar que más del 75% de los usuarios han elegido al menos dos puntos del centro para el patrón de desbloqueo seleccionado, incrementando así la seguridad del sistema.

Concluyendo finalmente que el sistema de contraseñas grafico propuesto, aumentaría la seguridad en los teléfonos inteligentes.

Summary

The idea of this assignment is to enhance the graphical password system currently used for the Google Android operating system, which has become one of the most popular graphical passwords nowadays. As a result of this popularity, the system has also become a target for different attacks searching for vulnerabilities. One of the vulnerabilities found, is based on the analysis of oily residue smudges on the touch screen to ascertain the pattern used to unlock the device. This attack is surprisingly simple and alarmingly effective.

The design of this proposed system is based on the daily use of smart phones, where a finger is used to interact with the device. These movements with the finger are typically focused at the center of the touch screen and therefore, the number of points in the center of the screen has been increased from one to four. With this idea, the hope is to hinder the attainment of the pattern used by the user. By using the center points, the oily residue smudges from the process of unlocking the device could get altered by the oily residue smudges on the screen from the device's daily use.

Along with the idea of increasing the number of points in the grid, a color is added to the pattern. This factor helps to increase the security, even if the user chooses a simple pattern without the central points in the grid. In the current system the attacker only had to guess the direction of the chosen pattern, but now the attacker also has to guess the chosen color from nine different possibilities.

A prototype for the Android platform has been created to evaluate the acceptation of the proposed system. The tests have been carried out with a group of voluntaries with different profiles. This group attempts to simulate a potential scope of real users for this system.

After analyzing the tests, we can conclude that the users didn't have problems remembering the chosen patterns, though they were more complex than the patterns from the current system. With these tests, we have also shown that more than 75% of the users have chosen at least two of the central points in their patterns, increasing the security of the system.

The final conclusion is that this proposed graphical password would increase the security of smart phones.

Índice

Introducción

1. Estado del arte de los sistemas de contraseñas gráficos

1.1 Sistemas recall-based

1.2 Sistemas recognition-based

1.3 Sistemas Cued-recall

1.4 Elección del sistema con el que se trabajara

2. Descripción general del nuevo sistema propuesto

2.1 Objetivo del sistema propuesto

3. Análisis del sistema propuesto

3.1 Elaboración de las máquinas de estados para el registro del patrón

3.2 Elaboración de las máquinas de estados para desbloquear el móvil

4. Diseño del sistema propuesto

4.1 Tipo de aplicación

4.2 Planificación del diseño

4.3 Funcionamiento del diseño

5. Detalles de la implementación del prototipo

5.1 Definición de interfaces de usuario

5.2 Descripción de las partes más importantes del código

5.3 Descripción de las partes del layout común entre las distintas actividades

6. Análisis de la seguridad

6.1 Espacio de contraseñas del sistema de desbloqueo de Android

6.2 Espacio de contraseñas del sistema propuesto

7. Evaluación de usabilidad con usuarios reales

7.1 Resumen de la interacción con los voluntarios para las pruebas

7.2 Análisis de los resultados obtenidos

8. Conclusiones

Bibliografía

Introducción

Una de las grandes lacras del omnipresente sistema de autenticación basado en nombres de usuarios y contraseñas es la dificultad que tienen los usuarios para recordar contraseñas seguras. Esto hace que a menudo se utilicen contraseñas simples, que resultan fáciles de adivinar con técnicas de ingeniería social o de romper con ataques de diccionario Otro problema, es la reutilización de la misma contraseña en distintas aplicaciones, para evitar memorizar varias. Una alternativa son las contraseñas gráficas, que de acuerdo a los estudios psicológicos realizados, el cerebro humano es capaz de recordar mejor la información visual que información textual.

Por estas razones, las contraseñas alfanuméricas tradicionales pueden sustituirse por nuevos sistemas de autenticación gráfica.

El objetivo de este trabajo es diseñar un mecanismo de autenticación gráfica, lo cual implica incorporar algún componente gráfico en los procesos de autenticación, para que el usuario seleccione una imagen, dibuje una forma o elija colores, en vez de introducir una contraseña alfanumérica.

1. Estado del arte de los sistemas de contraseñas gráficos

Siempre es importante conocer los sistemas existentes, pero en este caso es una parte fundamental, porque a mayor conocimiento se tenga de ellos, mejor será la elección de la mejora a realizar.

A continuación se enumeran las propuestas más relevantes de este tipo de esquemas.

1.1 Sistemas recall-based

En estos sistemas los usuarios típicamente dibujan sus contraseñas en lienzos en blanco o en una cuadricula. Algunos usuarios a veces idean formas de usar la interfaz como una pista, para poder recordar la contraseña con más facilidad.

Draw-A-Secret (DAS) [Jermyn et al. 1999] fue el primer sistema de contraseñas graficas propuesto. Donde un dibujo consiste en un trazo continuo o preferiblemente en varios trazos separados por “pen-ups”, secuencia que iniciaría el siguiente trazo en otra celda. Este sistema codifica el dibujo hecho por el usuario, usando la secuencia de coordenadas de la cuadricula por donde fue trazado, produciendo así una contraseña DAS. Su longitud será el resumen del número total de los pares de coordenadas de todos los trazos

BDAS [Dunphy and Yan 2007] similar al DAS, pero añadiendo una imagen de fondo para animar al usuario a crear contraseñas más complejas.

YAGP (Yet Another Graphical Password) [Gao et al. 2008] una modificación de DAS donde un dibujo se considera correcto cuando encaja en un cuadrante basándose en la distancia de cadenas de Levenshtein y los trazos se han hecho en la dirección correcta.

Passdoodle [Goldberg et al. 2002; Varenhorst 2004] es similar al DAS, permite al usuario crear un dibujo a mano, pero usando un proceso más complejo sin una cuadricula visible. Para añadir variabilidad en los dibujos a mano, se usan distintos colores y grosores del trazo y se tiene en cuenta la velocidad de trazado.

PassShapes [Weiss and De Luca 2008] las contraseñas son traducidas a caracteres alfanuméricos basándose en ocho direcciones de trazos reconocidas en intervalos de 45º. Durante la autenticación, la contraseña puede ser dibujada en diferente tamaño o lugar en la pantalla y aun así ser traducida correctamente, si las direcciones de los trazos realizados son precisas. El espacio de contraseña es reducido, porque solo es posible tener ocho elecciones con cada trazo.

Pass-Go system [Tao and Adams 2008] los usuarios dibujan sus contraseñas usando puntos de intersección de la cuadricula. Los movimientos de los usuarios son capturados en las líneas de la cuadrícula y las intersecciones, eliminando el impacto de pequeñas variaciones en el trazo. El espacio de contraseñas es mayor que en DAS porque permite los movimientos diagonales y también tiene un parámetro adicional con el color de los trazos.

GrIDsure [2009] un producto comercial que muestra dígitos en una cuadricula de 5x5. El usuario selecciona y memoriza un patrón que consiste en un subconjunto ordenado de una cuadricula de 25 cuadrados y la correspondiente introducción de dígitos dentro de ellos usando un teclado. En las subsiguientes autenticaciones, los dígitos son mostrados arbitrariamente dentro de las celdas de la cuadricula y los usuarios introducen la nueva secuencia de dígitos que se encuentran dentro de las celdas, de su patrón memorizado.

1.2 Sistemas recognition-based

Estos sistemas generalmente requieren que los usuarios memoricen un portafolio de imágenes durante la creación de la contraseña, para después tener que reconocerlas entre un grupo de otras imágenes y así recrear la secuencia registrada. Este sistema se basa en la excepcional habilidad que tiene el ser humano para reconocer imágenes que ha visto previamente.

Passfaces [Passfaces Corporation 2009] los usuarios preseleccionan un conjunto de caras humanas. Durante la autenticación un panel de caras candidatas es mostrada. Los usuarios deben seleccionar la cara que pertenece a su conjunto de entre los señuelos. Varias rondas como esta son repetidas con diferentes paneles. Para autenticarse con éxito, cada ronda debe de ser ejecutada correctamente. El conjunto de imágenes en un panel permanece constante entre autenticaciones, pero las imágenes son permutadas dentro de un panel, incurriendo así algún coste en usabilidad.

Story [Davis et al. 2004] fue propuesto como un sistema de comparación de caras. Los usuarios primero tienen que seleccionar una secuencia de imágenes de sus portafolios. Para autenticarse, a los usuarios se les muestra un panel de imágenes y deben identificar sus imágenes del portafolio de entre los señuelos. Story introduce un componente secuencial: Los usuarios deben seleccionar las imágenes en el orden correcto. Para ayudar a memorizar, los usuarios fueron instruidos para mentalmente construir una historia que relacionase las imágenes del día a día en su conjunto.

Déjà Vu [Dhamija and Perrig 2000] los usuarios seleccionan y memorizan un subconjunto de imágenes de “arte aleatorio” de un gran número de ejemplos para su portafolio. Para autenticarse los usuarios deben reconocer las imágenes pertenecientes a su portafolio predefinido de entre un conjunto de imágenes de señuelo. Imágenes de arte aleatorio son usadas para hacer más difícil que los usuarios puedan escribir sus contraseñas o compartirlas con otros usuarios al describir las imágenes usadas.

GPI (Graphical Password with Icons) y GPIS (Graphical Password with Icons suggested by the System) [Bicakci et al. 2009] en estos sistemas los usuarios se autentican seleccionando sus seis iconos en orden de un panel de 150 iconos. Estos sistemas se diferencian solo en como la contraseña es elegida. GPI permite a los usuarios elegir seis iconos cualesquiera como su contraseña. En GPIS, las contraseñas son sugeridas por el sistema, pero los usuarios pueden barajar varias hasta que encuentren una contraseña aceptable, reduciendo así problemas con la elección.

1.3 Sistemas Cued-recall

Estos sistemas típicamente requieren que los usuarios recuerden y enfoquen ubicaciones específicas dentro de una imagen. Esta característica intenta reducir la carga en la memoria de los usuarios, debido a que esta es una tarea más sencilla que la de memorizar una contraseña de texto por ejemplo. Estos sistemas se basan en identificar ubicaciones específicas. Hollingworth and Henderson [2002] demostraron que las personas recuerdan con exactitud detalles de objetos, que han previamente visualizado atentamente. Por lo cual el objetivo de este sistema es que el usuario sea capaz de recordar con precisión partes específicas de una imagen para usarlas como su contraseña.

PassPoints [Wiedenbeck et al. 2005] es el sistema dominante en este género. La contraseña es una secuencia de cinco puntos seleccionados (click-points) en cualquier parte de una imagen asignada por el sistema. El usuario selecciona los puntos haciendo un clic en cada uno de ellos usando el ratón. Durante la autenticación se debe repetir la secuencia en el orden correcto y de forma precisa, acorde con una tolerancia específica del sistema. La imagen ayuda a recordar la localización puntos seleccionados originalmente.

Robust discretization [Birget et al. 2006] y optimal discretization [Bicakci 2008] son posibles alternativas, donde se proponen distintos métodos para calcular la precisión con la que se deben aceptar los puntos seleccionados en la imagen. Esta discrecionalizacion hará variar el tamaño de la contraseña del lado del servidor para ser verificada.

Cued Click-Points (CCP) [Chiasson et al. 2007] es un esquema basado en la selección con un clic, donde el usuario selecciona un punto en cada una de las cinco imágenes mostradas en secuencia. Cada imagen se almacena con las coordenadas del punto donde se hizo el clic. Durante la autenticación, el usuario recibe una respuesta inmediata de si ha seleccionado un punto de forma errónea mediante la aparición de una imagen que no reconoce. En ese momento, el usuario podrá reiniciar el proceso autenticación para corregir su contraseña.

Persuasive Cued Click-Points (PCCP) [Chiasson et al. 2008] es una variación del diseño CCP para persuadir al usuario en la selección de puntos más aleatorios. Funcionalmente es como CCP, pero aquí durante la creación de la contraseña, la imagen mostrara un cuadrado sombreado posicionado en un área aleatoria, limitando la zona donde el usuario puede seleccionar el punto. El usuario podrá pulsar el botón “shuffle” para variar la localización del cuadrado sombreado a otra área aleatoria dentro de la misma imagen. En la secuencia de autenticación las imágenes mostraran su aspecto original sin las zonas sombreadas.

1.4 Elección del sistema con el que se trabajara

Se trabajará con el sistema Pass-Go, que forma parte de los sistemas recall-based. Concretamente el sistema que se pretende desarrollar es un sistema basado en Pass-Go que ha sido desarrollado comercialmente para desbloquear la pantalla en los teléfonos móviles que llevan el sistema operativo de Google Android.

Precisamente por haber sido desarrollado comercialmente para el sistema de Android, se ha convertido en uno de los sistemas de contraseñas graficas más extendido tanto en uso como en popularidad. Este sistema es responsable de proteger accesos de terceros a nuestro móvil, lo que significa en estos tiempos, proteger el acceso a nuestro correo personal y/o profesional, el acceso a nuestros contactos, posibles ataques de suplantación de identidad en redes sociales, acceso a documentos personales guardados localmente o en la nube (Cloud), etc.

Los casos mencionados anteriormente son algunas de las razones por las que este sistema se ha convertido en el objetivo de muchos ataques en busca de posibles vulnerabilidades. De las cuales hay que destacar la vulnerabilidad más peligrosa probada por el estudio de Aviv et al. 2010 “Smudge Attacks on Smartphone Touch Screens”. La investigación de Aviv et al. demuestra como un atacante puede determinar el patrón usado por el usuario, siguiendo las manchas o residuos de grasa que dejan los dedos en la pantalla.

En dicho estudio se ha llegado a demostrar que, con el material adecuado, se puede identificar parcialmente el patrón usado en el 92% de los casos y totalmente en el 68%.

Debido a este estudio, se ha elegido este sistema de contraseñas para ser mejorado y poder aportar mayor seguridad a los usuarios de Google Android.

2. Descripción general del nuevo sistema propuesto

Una vez explicada la importancia que tiene este sistema de desbloqueo en la sociedad actual, procederemos a describir el sistema propuesto en detalle. De esta manera llegaremos a obtener un mayor entendimiento de cómo funciona y que ventajas nos aporta.

2.1 Objetivo del sistema propuesto

Para poder definir este sistema propuesto, primero habría que aclarar cuál es el mayor problema del sistema actual. Este problema se puede resumir en tres razones principales: Lo sorprendentemente persistentes que son los residuos dejados en la pantalla, la dificultad de accidentalmente obscurecer o borrar dichos residuos al guardar el dispositivo en el bolsillo o al pasarle un paño a la pantalla y la última, lo fácil que es analizar esos residuos para obtener el patrón utilizado con solo una cámara y un ordenador.

Teniendo en cuenta que siempre que se use un sistema de contraseña grafico para desbloquear la pantalla va a dejar residuos en ella, lo que se pretende con el sistema propuesto es dificultar al máximo la posible extracción del patrón utilizado.

Para conseguir este objetivo se aumenta el número de puntos en la cuadricula y se añade un nuevo factor, un color. Esta propuesta se crea basándonos en el uso cotidiano de los teléfonos con el sistema Android, donde la pantalla táctil sustituye al ratón y para utilizarla tenemos que deslizar al menos un dedo por la pantalla. Dichos movimientos con el dedo normalmente se hacen en el centro de la pantalla, lo que nos lleva a concluir, que los movimientos más habituales en la pantalla táctil se concentran en el centro de la cuadricula 4x4.

Teóricamente, si el patrón utilizado incluye varios de los puntos situados en el centro de la cuadricula (6, 7, 10 y 11 de la figura 1), conseguiríamos que parte del patrón quedara justo dentro de la zona de mayor uso del usuario. Con esto concentraríamos mayor cantidad de residuos en esta zona, dificultando así la obtención del patrón utilizado. Anteriormente en esa zona solo se encontraba un punto y era fácil de deducir la continuidad del patrón por los residuos en los puntos que lo rodeaban, pero ahora podrían quedar camuflados los residuos en cuatro de los dieciséis puntos de la cuadricula, dejando al atacante en un escenario más complejo.

En caso de que el usuario no utilice los puntos situados en el área de mayor uso de la pantalla táctil, el atacante aun tendrá que adivinar el color utilizado en el patrón.

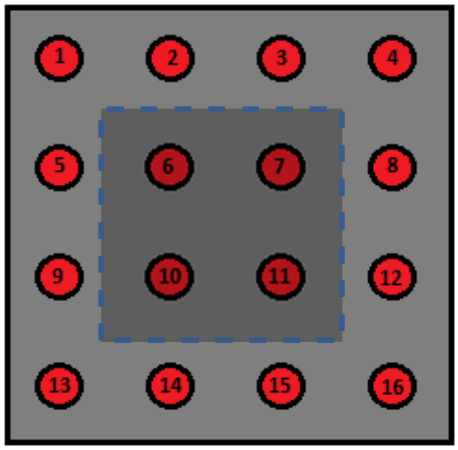


Figura 1

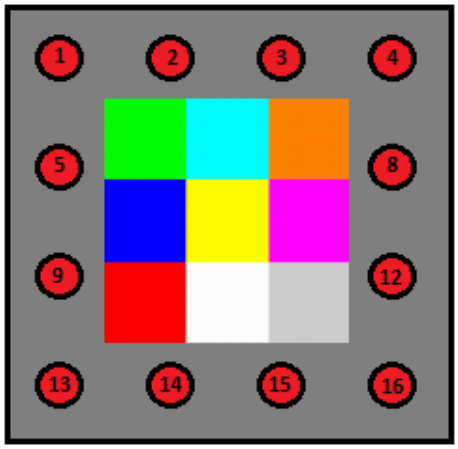


Figura 2

El sistema que se propone implica la modificación de la cuadricula de 3x3 a una de 4x4. Para seguir forzando al usuario a que no pueda crear una contraseña con una sola línea recta y que al menos tenga que realizar un cambio de dirección, la selección mínima de puntos para registrar la contraseña pasa de 4 a 5. Con este incremento de puntos en la cuadricula, se ve incrementado el número máximo de puntos seleccionables, pasando a ser ahora 16, que unido al parámetro adicional de la selección de un color en los trazos, aumentan considerablemente el espacio de contraseñas y por consiguiente su seguridad.

Para añadir mayor complicación a los atacantes se les añade la elección de un color antes de seleccionar la traza de puntos. Los posibles colores a seleccionar serán nueve y se mostraran en una paleta de colores, que cambiara aleatoriamente el orden de los mismos (figura 2). El color se podrá elegir con un simple toque sobre la paleta o arrastrando el dedo por la pantalla hasta que toque un color. Al seleccionar un color, la paleta desaparecerá de la pantalla, mostrando únicamente la cuadricula de puntos.

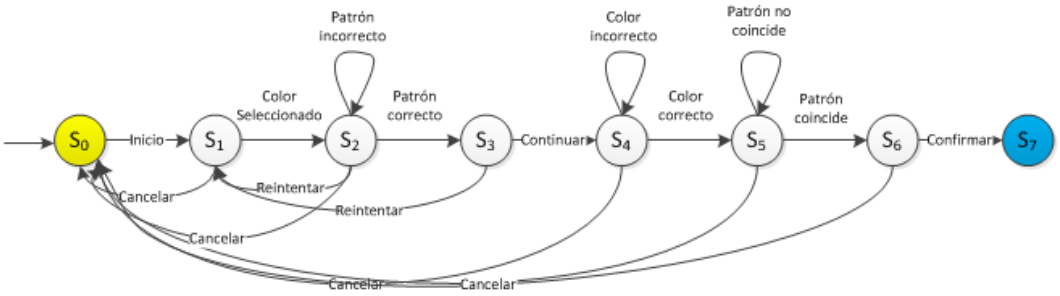
Con este nuevo parámetro, si el atacante consiguiese averiguar parcialmente la secuencia de puntos seleccionada por el usuario, aun tendría que averiguar el color elegido y aunque la elección del color dejase residuos no sería de utilidad, ya que los colores se muestran en posiciones aleatorias. Esto unido a que cuando se introduce una combinación errónea para desbloquear la pantalla, el sistema no especificara si se produjo el error en la selección del color, en la secuencia de puntos o en ambos. Dejando al atacante con un gran número de combinaciones para probar.

3. Análisis del sistema propuesto

Para el análisis del diseño se crearan varias máquinas de estados. Con estas máquinas se pueden contemplar todas las acciones permitidas en nuestro diseño, permitiéndonos tener mayor control y conocimiento del comportamiento de la aplicación.

3.1 Elaboración de las máquinas de estados para el registro del patrón

En la máquina de estados para el registro del patrón, se muestra el comportamiento de la aplicación sin entrar en detalles de cómo se registrara la secuencia de puntos.



El estado S0 representa el estado de la aplicación antes de iniciarse. Al iniciarla pasara automáticamente al estado S1.

El estado S1 representa la pantalla inicial de la aplicación, donde aparecerá la paleta de colores sobre la cuadricula de 4x4. Desde aquí se puede cancelar el registro y se volvería al estado S0. Si se selecciona un color pasara al estado S2.

El estado S2 representa la cuadricula de 4x4. Desde aquí se puede reintentar la elección del color y se volvería al estado S1. Si se selecciona una secuencia no permitida, nos quedamos en el mismo estado para realizar otra secuencia. Si se selecciona una secuencia permitida, se avanza al estado S3.

El estado S3 mostrara la misma pantalla que el estado anterior, pero con el trazo dibujado en la pantalla a la espera de confirmar la secuencia. Si se selecciona la opción de continuar, se registra la combinación y se avanza al estado S4. Si se selecciona la opción reintentar, se volvería al estado S1.

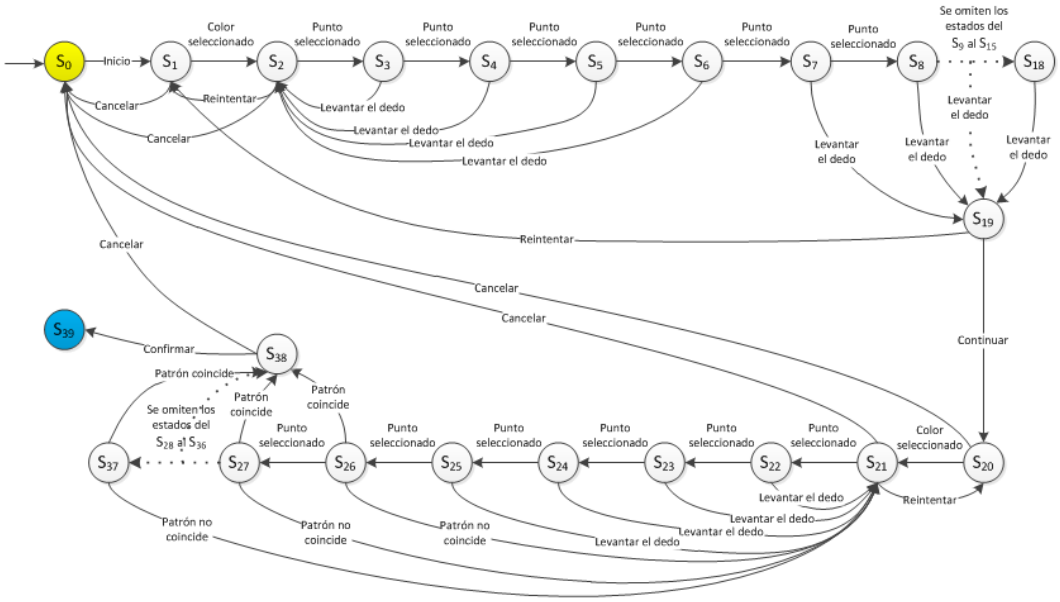
El estado S4 es muy similar al estado S1, pero se diferencian en que si la selección del color no coincide con la elección previa, seguirá en el mismo estado a la espera del color correcto.

El estado S5 es muy similar al estado S2, pero se diferencian en que si la selección de la secuencia no coincide con la selección previa, seguirá en el mismo estado a la espera de la secuencia correcta.

El estado S6 es muy similar al estado S3, pero ahora la aplicación te la opción de confirmar la secuencia. Si se selecciona confirmar se registrara el patrón completo y se avanzara al estado final.

El estado S7 es el estado final de la máquina y representa el final de la aplicación.

En la siguiente máquina de estados, se muestra el comportamiento de la aplicación en detalle de cómo se registrara la secuencia de puntos.



En esta máquina solo se explicaran los estados más importantes, ya que debido a su gran número, se convertiría en una explicación muy extensa.

Los estados S0 y S1 coinciden con la maquina anterior.

En el estado S2 se permite desplazar el dedo por la pantalla, hasta que pase por encima de un punto. Al capturar ese evento se registrara el primer punto y se pasara al siguiente estado.

Los estados del S3 al S6 muestran lo que pasaría si se levanta el dedo durante la selección y el número de puntos es menor que cinco. Al estar establecido un número mínimo de cinco, se volverá directamente al estado S2 para empezar de nuevo la selección.

En los estados del S7 al S18 al levantarse el dedo ya habrá una secuencia de puntos seleccionada mayor o igual a cinco, que sería una secuencia valida y nos llevaría al estado S19.

En el estado S19 se muestra el trazo dibujado en la pantalla a la espera de confirmar la secuencia. Si se selecciona la opción de continuar, se registra la combinación y se avanza al estado S20. Si se selecciona la opción reintentar, se volvería al estado S1.

El estado S20 es muy similar al estado S1, pero se diferencian en que si la selección del color no coincide con la elección previa, seguirá en el mismo estado a la espera del color correcto.

El estado S21 funciona igual que el estado S2.

Los estados del S22 al S25 tienen la misma funcionalidad que los estados del S2 al S6.

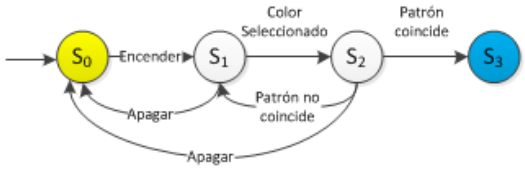
En los estados del S26 al S37 al levantarse el dedo ya habrá una secuencia de puntos seleccionada mayor o igual a cinco. Si coincide con la seleccionada anteriormente, se avanzara al estado S38. Si no, se volverá al estado S23 para volver a repetir la secuencia.

El estado S38 es muy similar al estado S19, pero ahora la aplicación te la opción de confirmar la secuencia. Si se selecciona confirmar se registrara el patrón completo y se avanzara al estado final.

El estado S39 es el estado final de la máquina y representa el final de la aplicación.

3.2 Elaboración de las máquinas de estados para desbloquear el móvil

En la máquina de estados para desbloquear el móvil se muestra el comportamiento de la aplicación sin entrar en detalles para tener una visión sencilla de su funcionamiento.



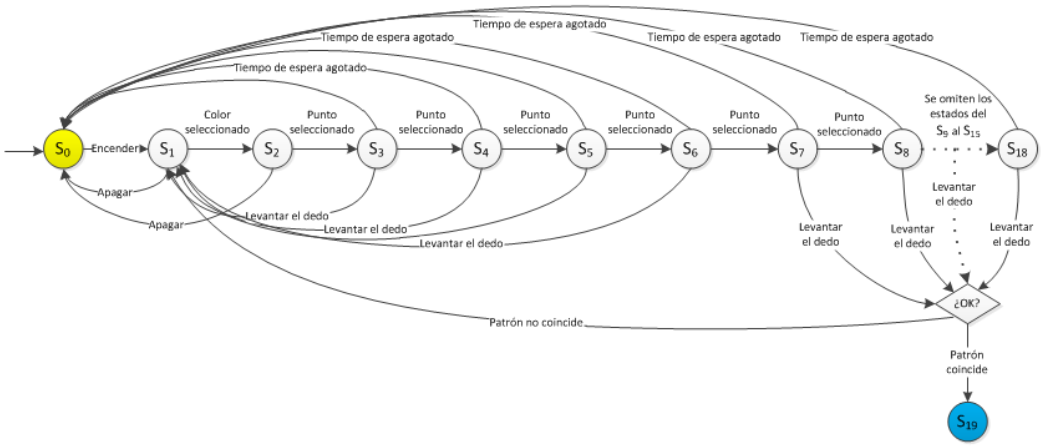
El estado S0 representa el estado de la aplicación antes de iniciarse, que equivaldría a tener el móvil con la pantalla desactivada. Al encender la pantalla, se inicia la aplicación y pasa automáticamente al estado S1.

El estado S1 representa la pantalla inicial de la aplicación, donde aparecerá la paleta de colores sobre la cuadricula de 4x4. En este estado si se apaga la pantalla, se volvería al estado S0, y si se selecciona un color pasaría al estado S2.

El estado S2 representa la cuadricula de 4x4. En este estado se tendrá que seleccionar la secuencia que coincida con la anteriormente registrada. Si no coincide la secuencia seleccionada o el color seleccionado en el estado S1 o ambos, volvemos al estado S1. Pero, Si el patrón coincide se avanza al estado S3.

El estado S3 es el estado final de la máquina y representa que la aplicación término con éxito. Lo que equivale a que el móvil se desbloqueó.

En la siguiente máquina de estados, se muestra el comportamiento de la aplicación más en detalle. Obteniendo un mejor entendimiento de cómo se comprueba que el patrón introducido coincide con el registrado para desbloquear el móvil.



El estado S0 representa el estado de la aplicación antes de iniciarse, que equivaldría a tener el móvil con la pantalla desactivada. Al encender la pantalla, se inicia la aplicación y pasa automáticamente al estado S1.

En los estados del S1 al S18 si se apaga la pantalla o se supera el tiempo de inactividad, se apagara la pantalla y nos devolverá al estado S0.

El estado S1 representa la pantalla inicial de la aplicación, donde aparecerá la paleta de colores sobre la cuadricula de 4x4. En este estado si se selecciona un color pasara al estado S2.

El estado S2 muestra la cuadricula de 4x4. En este estado se permite desplazar el dedo por la pantalla, hasta que pase por encima de un punto. Al capturar ese evento se registrara el primer punto y se pasara al siguiente estado.

Los estados del S3 al S6 muestran lo que pasaría si se levanta el dedo durante la selección y el número de puntos es menor que cinco. Al estar establecido un número mínimo de cinco, se descartara el patrón introducido y volverá directamente al estado S1 para empezar de nuevo la selección.

En los estados del S7 al S18 al levantarse el dedo ya habrá una secuencia de puntos seleccionada mayor o igual a cinco, que sería una secuencia valida. En este momento la aplicación tiene que examinar si el color y la secuencia de puntos seleccionados coinciden con el patrón previamente registrado. Si coincide nos llevaría al estado S19, pero si no coincide se volverá al estado S1 para volver a intentarlo.

El estado S19 es el estado final de la máquina y representa que la aplicación término con éxito. Lo que equivale a que el móvil se desbloqueó.

4. Diseño del sistema propuesto

Para empezar con el diseño de este sistema, lo primero que tenemos que plantearnos es que tipo de aplicación vamos a crear y cuál es la mejor manera de traducir las máquinas de estados que hemos definido anteriormente a nuestra aplicación.

4.1 Tipo de aplicación

La aplicación será diseñada para ser ejecutada en un sistema Android, pero no se podrá usar para sustituir al sistema actual de desbloqueo. Esta aplicación estará orientada al estudio de la usabilidad con distintos usuarios. Para su desarrollo se usara un lenguaje de programación orientado a objetos y su ejecución se basara en eventos. Estos eventos se capturaran y se procesaran acorde a las especificaciones definidas en las máquinas de estado.

4.2 Planificación del diseño

Lo primero que se define es la interfaz de la aplicación. Para nuestra aplicación será necesario crear una cuadricula y definir la ubicación de los objetos que harán la función de los puntos y la paleta de colores.

La paleta de colores será un objeto cuadrado, que estará subdividido en nueve cuadrados, los cuales representaran los nueve colores disponibles para el patrón. Este objeto se mostrara sobrepuesto a la cuadricula.

Los 16 puntos estarán representados por 16 objetos de forma circular y estarán distribuidos uniformemente en la cuadricula, formando una estructura de 4x4.

Los eventos capturados controlaran las siguientes acciones, cuando se posicione el dedo sobre la pantalla, el desplazamiento de este a través de la pantalla y cuando se levante el mismo de la pantalla. También se controlara si los botones han sido pulsados.

Cada evento que se capture tendrá que ser procesado, acorde con el estado en que se encuentra la aplicación.

Esta aplicación está dividida en dos partes principales, el registro del patrón y el uso del mismo para desbloquear la pantalla.

4.3 Funcionamiento del diseño

Esta aplicación simula el comportamiento del sistema propuesto para el sistema Android. En orden de poder usar la aplicación para desbloquear el móvil, primero se tendrá que registrar el patrón. Para poder probar los distintos comportamientos, nuestra aplicación nos dará cuatro opciones al principio. Registrar el patrón, desbloquear la pantalla, borrar el patrón registrado y salir de la aplicación (Figura 3).

Para poder simular el desbloqueo, se habrá tenido que registrar el patrón previamente, si no hay un patrón registrado, la aplicación nos avisara de ello y nos devolverá a la pantalla principal. Si se quisiese cambiar el patrón elegido, nos valdría con volver a seleccionar Registrar, sin tener que verificar que conocemos el patrón anterior. La opción de borrar el patrón, se encargará de eliminar el fichero donde se guarda el registro codificado del patrón.

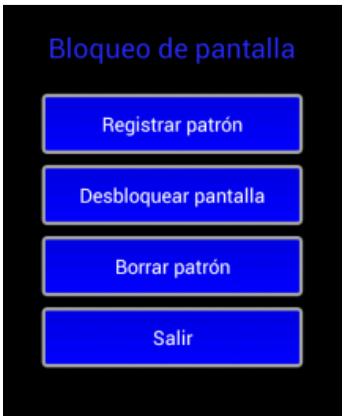


Figura 3

Una vez se inicie el proceso de registrar el patrón o desbloquear la pantalla, nos encontraremos con una primera pantalla donde se mostrará la paleta de colores sobrepuesta a la cuadricula de 4x4, esta parte es común para ambas. Para la interfaz de registrar el patrón se añadirán dos botones, para poder realizar las acciones de cancelar, continuar o reintentar (Figura 4). La paleta según se seleccione un color desaparecerá, dejando a la aplicación esperando para capturar la secuencia de puntos que se van a seleccionar. Los botones cambiarán de nombre y funcionalidad, dependiendo del estado en que se encuentre la aplicación. El botón continuar estará desactivado cuando no se permita su uso.

Para seleccionar el color se tiene que tocar con el dedo sobre alguno de los colores. Ya sea pulsando directamente sobre el color o arrastrando el dedo hasta llegar a un color. Cuando esto ocurra, la aplicación capturará el evento de que uno de sus objetos ha sido seleccionado y se guardará la información necesaria como primer dato del patrón. Después el objeto de la paleta se quitará de la cuadrícula.

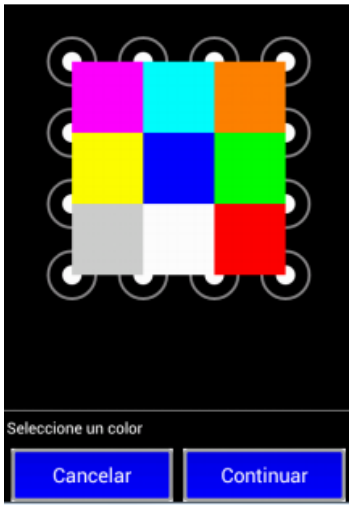


Figura 4

Después de seleccionar el color, la aplicación estará lista para registrar la secuencia de puntos. Este proceso se activará al tocar nuevamente la pantalla, pero no empezará a mostrar el trazo elegido por el usuario hasta que se haya seleccione el primer punto.

Cada vez que el dedo pase por encima de otro punto, se capturara el evento y se comprobará si dicho punto ya ha sido seleccionado. Si aún no ha sido seleccionado, se comprueba si hay más puntos entre el punto anterior y el actual, que tampoco hayan sido seleccionados para seleccionarlos también y añadirlos en la secuencia. Si el punto ya había sido seleccionado, no se realiza ninguna acción.

Cuando se levante el dedo de la pantalla, la aplicación capturará el evento y procesará la información de la secuencia almacenada. Si la longitud de esta secuencia es menor que cinco, se descartara. Si es mayor o igual a cinco, la aplicación procesará los datos almacenados.

El procesamiento de los datos almacenados dependerá en el estado que se encuentre la aplicación. Si se está registrando el patrón, lo dará por válido y esperará a que el usuario repita el patrón completo de nuevo para poder registrarlo. Si se está desbloqueando el móvil, se comprobará si el patrón introducido coincide con el patrón registrado. En caso de que coincida el patrón se desbloquearía la pantalla, pero si no coincide, se devolvería automáticamente a la pantalla inicial para reintentar el patrón.

Para ayudar al usuario a recordar el color elegido, se ha intentado crear una interfaz amigable. Donde la paleta de colores en el proceso de confirmación del registro vuelve a mostrarse exactamente en el mismo orden que cuando se eligió el color por primera vez y también utilizando ese mismo color para dibujar el trazo de la secuencia de puntos.

5. Detalles de la implementación del prototipo.

Estudiando los posibles entornos de programación disponibles que cubren nuestros requerimientos para realizar este prototipo, el que mejor se ajusta y por tanto el que se ha usado es el paquete ADT Bundle propuesto por Android. Este paquete incluye todo lo necesario para desarrollar nuestro prototipo, incluido un emulador ajustable para hacer pruebas con distintas configuraciones de pantalla.

5.1 Definición de las interfaces de usuario

En el apartado anterior (Funcionamiento del diseño) ya se ha dado una breve descripción de la interfaz de usuario de esta aplicación, por lo tanto en este punto, definiremos la interfaz de usuario más técnicamente.

La aplicación tiene cuatro actividades principales, la actividad del menú principal, dos para el registro del patrón y una para desbloquear la pantalla.

La actividad del menú principal se encarga de lanzar las actividades para registrar el patrón y desbloquear la pantalla. Además, desde esta actividad podemos borrar el archivo donde se almacena el hash de la contraseña y finalizar la aplicación.

Las actividades para registrar el patrón se dividen en dos, en la primera se podrá seleccionar cualquier patrón, siempre y cuando el número de puntos seleccionados sea igual o mayor que cinco. Una vez seleccionado este primer patrón podremos elegir entre reintentar, lo que nos devolvería al comienzo de esta misma actividad o continuar, lo que haría que se registre temporalmente el patrón seleccionado en la memoria interna del móvil y se lanzase la segunda actividad del registro.

En esta segunda actividad la aplicación mostrará un escenario similar al anterior, pero ahora la aplicación comparará el patrón que se seleccione, con el patrón previamente seleccionado en la actividad anterior. Si no se introdujese el mismo patrón, la aplicación asume que se ha producido un error al seleccionar los puntos y limpiará automáticamente el trazo introducido para que se pueda volver a repetir la selección de puntos. Si el error se hubiese producido al confirmar el color, se tendrá que seleccionar la opción reintentar para que nos vuelva a mostrar la paleta de colores. Estas acciones se podrán repetir de manera indefinida hasta que el usuario consiga confirmar correctamente el patrón previamente seleccionado o hasta que cancele la acción de registrar un patrón.

Una vez introducido el patrón correctamente, la aplicación nos dará la opción de confirmarlo o cancelar. Si confirmamos, se terminaría esta actividad, dejando el patrón seleccionado como el patrón configurado para desbloquear la pantalla en nuestra aplicación. Si por el contrario seleccionamos cancelar, también se saldría de la actividad, pero se borraría el fichero donde se había guardado temporalmente el hash del patrón anteriormente seleccionado y nos quedaríamos sin ningún patrón registrado.

La actividad de desbloquear la pantalla es similar a la actividad de registrar el patrón, pero en esta actividad no habrá opciones. Simplemente se mostrara la paleta de colores sobre la cuadricula de 4x4 a la espera que se introduzca el patrón anteriormente registrado. Si no se seleccionase el patrón correctamente, la actividad se reiniciaría automáticamente, volviendo al estado original para que se vuelva a introducir nuevamente el patrón, sin dar ninguna información de que fallo en el intento anterior. Si se introduce el patrón correcto, la aplicación nos informará que el patrón seleccionado es el correcto y terminará esta actividad.

5.2 Descripción de las partes más importantes del código

Teniendo en cuenta que las actividades usadas para el registro del patrón y para desbloquear la pantalla son bastante similares, se ha realizado una implementación lo más modular posible para poder reutilizar los métodos diseñados en las distintas actividades.

Para poder realizar esto se ha creado una clase llamada Procesador, que es la que se encarga entre otras cosas de inicializar las matrices que representan la cuadricula de 4x4 y la paleta de colores con sus coordenadas relativas al tamaño de la pantalla, de examinar si los puntos seleccionados habían sido previamente seleccionados o no, y si existen puntos en medio que deberían o no añadirse a la secuencia.

Precisamente, el análisis de los posibles puntos existentes entre los seleccionados y si estos se deben de añadir a la secuencia o no, es una de las funciones más importantes de la clase Procesador y la mostramos a continuación.

```
private void PuntosMedio(int punto) {
int[] coordenadas;

//Si hay menos de 2 puntos seleccionados finaliza
if (secuencia[0] < 2)
       return;
else
       //getCoordenadas devuelve las coordenadas del punto actual y del anterior
       coordenadas = getCoordenadas(punto);

//Calcula si la linea creada entre el punto actual y el anterior
//forman una fila, una columna o una diagonal:
//Valor absoluto para comprobar si los puntos son correlativos
int x_x = Math.abs(coordenadas[0] - coordenadas[2]);
int y_y = Math.abs(coordenadas[1] - coordenadas[3]);

//Si X1=X2 y las coordenadas de las Y no son correlativas se analiza la columna
if ((coordenadas[0] == coordenadas[2]) && (y_y >= 2)) {
       columna (coordenadas[0], coordenadas[1], coordenadas[3]);

//Si Y1=Y2 y las coordenadas de las X no son correlativas se analiza la fila
} else if ((coordenadas[1] == coordenadas[3]) && (x_x >= 2)) {
       fila (coordenadas[1], coordenadas[0], coordenadas[2]);

//Se comprueba la diferencia entre las coordenadas para asegurar que no son correlativas
//También se descartan las diagonales no permitidas, para ello se comprueba que la
//diferencia entre X1 y X2 es igual a la diferencia entre Y1 e Y2 y
//que las sumas de las diferencias sean iguales a 4 o 6 confirma que son permitidas
} else if (((x_x + y_y == 4) || (x_x + y_y == 6)) && (x_x == y_y))
       diagonal(coordenadas[0], coordenadas[1], coordenadas[2],coordenadas[3]);
}
```

El método PuntosMedio se apoya en los métodos fila, columna y diagonal para analizar en cada caso si existen puntos intermedios. Estos tres casos se tratan de manera separada porque aun siendo similares, son casos distintos. En la fila la coordenada Y será constante y en la columna lo será la coordenada X, mientras que en la diagonal varían las dos.

Para poder explicar el principio básico de este análisis se usara como ejemplo la selección del punto 1 y el punto 16, con las coordenadas (x,y) siguientes (1,4) y (4,1) respectivamente. Donde suponemos que los puntos intermedios 6 y 11 con las coordenadas (2,3) y (3,2) respectivamente, aún no han sido seleccionados (Figura 5).

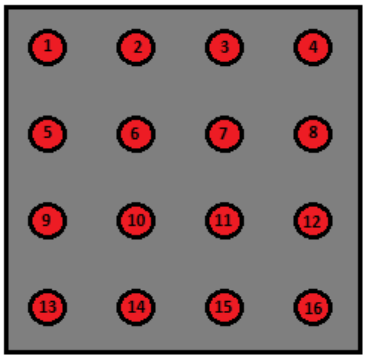


Figura 5

Una vez se han obtenido las coordenadas de los puntos seleccionados, lo primero que se hace es calcular las diferencias absolutas de las X y las Y. En este caso el resultado sería $|X_1| - |X_2| = |1| - |4| = 3$ y $|Y_1| - |Y_2| = |4| - |1| = 3$. Después se comprueba si las coordenadas X o Y son iguales, pero como tanto las X como las Y son distintas, deducimos que estamos tratando una diagonal.

Los tipos de las diagonales que nos podemos encontrar son distintos, pero solo analizaremos las que pertenecen al grupo de las llamadas permitidas, que son las que pueden tener puntos intermedios. Por ejemplo entre el punto 1 y el 16 están los puntos 6 y 11. Sin embargo entre los puntos 1 y 15 no hay puntos en medio, otro ejemplo de diagonal no permitida es entre 1 y 6, ya que tampoco existen puntos intermedios.

Una vez explicada la diferencia entre diagonales permitidas y no permitidas, procedemos a explicar cómo podemos determinar cuáles son las permitidas. Lo primero es comprobar que el valor de la diferencia absoluta entre las X sea igual a la diferencia absoluta entre las Y. Después comprobamos que la suma de las diferencias absolutas entre las X y las Y son igual a 4 o 6. Lo cual nos indicaría que puede haber uno o dos puntos intermedios.

Para calcular estos puntos intermedios se usa el método diagonal, que mostramos y explicamos a continuación.

```
private void diagonal(int x1, int y1, int x2, int y2) {
// Se calculan las coordenadas de los puntos intermedios
// Por ejemplo de (1,4) a (4,1) se busca conseguir el (2,3)
if (x1 > x2)
       x1--;
else
       x1++; // Se le suma 1 a la X1 --> 2
if (y1 > y2)
       y1--; // Se le resta 1 a la Y1 -->  3
else
       y1++;
// Si el (1,4) inicial ha llegado al (4,1) final se finaliza
if ((x1 == x2) && (y1 == y2))
       return;
else
//Comprueba si el punto que coincide con las nuevas coordenadas ha sido seleccionado
       existePunto(matrix[x1][y1]);
// Se llama recursivamente hasta que las coordenadas sean iguales
diagonal(x1, y1, x2, y2);
       return;
}
```

Basándonos nuevamente en el ejemplo de la selección del punto 1 al punto 16, nos encontramos que desde las coordenadas (1,4) a (4,1) tenemos las coordenadas intermedias de (2,3) y (3,2) que pertenecen a los puntos 6 y 11 respectivamente. Lo que este método hace es calcular las coordenadas siguientes a (1,4) en la diagonal, comprueba si el punto 6 ha sido seleccionado anteriormente usando el método existePunto (Este se encargará de añadirlo a la secuencia si no había sido seleccionado anteriormente), después comprueba que quedan puntos intermedios entre las nuevas coordenadas X_1 e Y_1 (2,3) y X_2 e Y_2 (4,1), si existe otro punto como sucede en nuestro ejemplo, se vuelve a llamar recursivamente para hacer la misma operación. En caso de no haber más puntos, simplemente terminaría el método.

Otra parte importante del código es la utilizada para detectar cuando el usuario selecciona un punto. Para realizar esto, lo primero que hacemos es implementar el método onTouchEvent para que capture los eventos producidos al tocar la pantalla. De esta manera cada vez que se toque la pantalla, se desplace el dedo por ella o se deje de tocar el método onTouchEvent capturara el evento producido y podremos obtener las coordenadas (X,Y) donde se produjo la acción.

Una vez que tenemos las coordenadas podemos calcular si la acción se encuentra dentro de las coordenadas donde hemos dibujado los puntos. Para poder calcular esto usamos el método SobrePunto(float x, float y) que recibe las coordenadas de (X,Y). A continuación se muestra el código de este método.

```
private boolean SobrePunto(float x, float y) {
//Iteración encargada de conseguir los límites de los 16 puntos
for (int i = 0; i < 16; i++) {
	//Aumenta la sensibilidad con respecto el método calculaCordenadas
	limites = brain.calculaCordenadasSensible(i, ancho, largo);
	//Si las coordenadas x,y están dentro de los limites, están sobre el punto i+1
	if ((x >= limites[0] && y >= limites[1])&& (x <= limites[2] &&
		y <= limites[3] && conPaleta == false)) {
		//Comprueba si el punto actual ya ha sido seleccionado, si no lo añade
		if (brain.ProcesarPunto(i + 1)) {
			//Se calcula el centro del punto para dibujar el trazo
			finX = mX = (limites[0] + limites[2]) / 2;
			finY = mY = (limites[1] + limites[3]) / 2;
			//Vector que almacena los puntos recién seleccionados
			modificados = brain.getModificados();
			for (int j = 1; j <= modificados[0]; j++) {
			//Se le cambia el color a los puntos seleccionados
			contorno[modificados[j] - 1].getPaint().setColor(color);
			}
		return true;
		}
	}
}
return false;
}
```

Como se puede observar, se utiliza un FOR para calcular los límites de los 16 puntos si fuese necesario. Ya que si las coordenadas se localizan antes de llegar al último punto se detiene el proceso de búsqueda. Si las coordenadas han podido ser localizadas, se llama al método ProcesarPunto(int punto) donde se le pasa como parámetro el punto donde se han localizado las coordenadas. Este método se encarga de comprobar si el punto ya ha sido seleccionado previamente o no. Si el punto no ha sido seleccionado previamente, se selecciona y se busca si existen puntos intermedios entre el punto que se acaba de seleccionar, con el anterior (Si es el primer punto que se selecciona, se trata de forma distinta).

5.3 Descripción de las partes del layout común entre las distintas actividades

Teniendo presentes que las actividades de registrar un patrón y desbloquear la pantalla son bastante similares, era de esperar que sus respectivos layout también lo fuesen. A continuación se muestra la apariencia del layout grafico (Figura 6) y parte del su código XML.

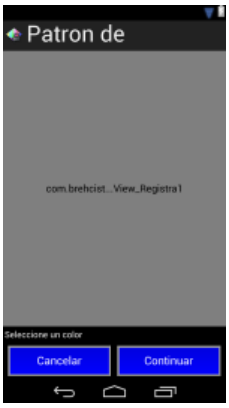


Figura 6

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    ...
    android:orientation="vertical" >

    <view
        ...
        class="com.brehcist.patrondedesbloqueo.View_Registra1" />

    <TextView ... />

    <TableLayout ... >

        <TableRow ... >

            <Button ... />

            <Button ... />

        </TableRow>

    </TableLayout>

</LinearLayout>
```

Lo importante de este fichero XML es el campo View, donde indicamos que queremos usar una vista personalizada ubicada en la clase "com.brehcist.patrondedesbloqueo.View_Registra1". Esta clase se implementa como una extensión de la clase view predefinida por el sistema y contiene los métodos para tratar los componentes gráficos usados en nuestra actividad.

```
public View_Registra1(Context cxt, AttributeSet attrs) {
       super(cxt, attrs);
       ...
}
```

Como podemos ver en el constructor utilizado hay dos parámetros, el primero de tipo Context te permite acceder al contexto de aplicación, por ejemplo para utilizar recursos de esta aplicación y el segundo de tipo AttributeSet, que te permite acceder a los atributos de esta vista, cuando sea creada desde XML.

Android realiza un proceso de varias pasadas para determinar el ancho y alto de cada vista dentro de un Layout. Cuando finalmente ha establecido las dimensiones de una vista llamará a su método onSizeChanged(). Este nos indica los parámetros del ancho y alto asignado.

El método onDraw() lo tenemos que sobrescribir ya que es el utilizado por el sistema para dibujar las vistas de los componentes gráficos que son mostrados en la pantalla. Para poder dibujar algo se necesitan cuatro componentes básicos: Un bitmap que contenga los pixels, un canvas para alojar los componentes escritos dentro del bitmap, un método que nos permita dibujar gráficos dinámicamente mediante primitivas para representar líneas, círculos, texto, etc. Y por último, un paint, que es el pincel con el que podremos definiremos el color, grosor de trazo, transparencia, etc.

Este método se llamara cada vez que se va a dibujar en la pantalla, por lo cual forzamos su llamada con el método invalidate() cada vez que capturamos un moviendo del dedo en la pantalla. De esta manera podemos dibujar en tiempo real el trazo realizado por el usuario, usando como coordenadas de origen el último punto seleccionado y como coordenadas de destino, las coordenadas capturadas con la nueva posición del dedo en la pantalla.

6. Análisis de la seguridad

En este apartado se realizara la comparación del espacio de contraseñas del sistema actual usado por Android, con el del sistema propuesto. Se podrá observar que con los cambios propuestos se mejora mucho la seguridad de este sistema, aunque solo se use la combinación más corta permitida.

6.1 Espacio de contraseñas del sistema de desbloqueo de Android

En el espacio de contraseñas de la versión actual tenemos que tener en cuenta que la combinación más pequeña permitida por el sistema es de cuatro puntos y equivale a 1624 posibilidades permitidas. La combinación más grande permitida es de nueve puntos y equivale a 140704 posibilidades permitidas.

Para calcular las posibilidades permitidas hay que tener en cuenta, que un punto no se puede seleccionar más de una vez en cada combinación y que si existe un punto en medio de los seleccionados que no se hubiese seleccionado previamente, se tendrá que incluir en la selección. Para calcular las combinaciones posibles sin repetición se puede usar la formula siguiente:

$C_{(m,n)} = m! / (m – n)!$

Si la aplicamos al número máximo de puntos seleccionables, obtendremos 9! / (9 – 9)! = 9! = 362880 combinaciones sin repetición, de las cuales solo 140704 están permitidas.

Mirando un poco más en profundidad, podemos observar las distintas combinaciones posibles y sus respectivas posibilidades permitidas.

4 puntos: 1624

5 puntos: 7152

6 puntos: 26016

7 puntos: 72912

8 puntos: 140704

9 puntos: 140704

Todas estas combinaciones suman un total de 389112 combinaciones permitidas, que equivale al espacio total de posibles claves.

6.2 Espacio de contraseñas del sistema propuesto

El espacio de contraseñas del sistema propuesto es superior que el del sistema actual y lo podemos apreciar desde la combinación más pequeña aceptada por el sistema, que es de cinco puntos y tiene 154680 posibilidades permitidas. Esto añadido al color que se ha de seleccionar también, lo aumentan a 1392120. Lo que equivale a 1390496 posibilidades más que la secuencia más pequeña permitida por el sistema actual.

Para poder calcular las combinaciones permitidas para los 16 puntos no existe una formula sencilla, por lo cual se ha tenido que desarrollar un programa que sea capaz de crear y analizar todas las combinaciones posibles sin repeticiones, obteniendo así solo el número de combinaciones permitidas.

Debido al gran número de combinaciones posibles cuando se seleccionan 16 puntos, se ha tenido que cambiar la aproximación original, ya que el programa se veía obligado a analizar 20922789888000 combinaciones. Esta tarea no es imposible, pero requiere de muchos días de cálculo para cualquier ordenador, dejándola como una opción inviable.

Para conseguir mejorar el código, hay que estudiar la simetría que hay en la cuadricula de 4*4. Como se puede apreciar en la imagen siguiente (Figura 7), hay casos comunes. Ya que el número de combinaciones permitidas que comiencen con el punto número 1, serán iguales que el número de combinaciones permitidas que empiecen con 4, 13 y 16. De la misma manera encontramos simetría con el número de combinaciones permitidas que empiecen con 6, 7, 10 y 11. También hay simetría y vuelven a coincidir el número de combinaciones permitidas que empiecen con 2, 3, 5, 8, 9, 12, 14 y 15.

Por lo tanto, viendo que hay tres grupos de puntos que representan todos los casos, podemos simplificar el código para calcular solo las combinaciones permitidas que empiecen con 1, 2 y 6. Los resultados obtenidos que comiencen con 1 y 6 se han de multiplicar por 4 y los obtenidos por 2 se han de multiplicar por 8. De esta manera obtendremos todas las combinaciones posibles permitidas en un tiempo aceptable.

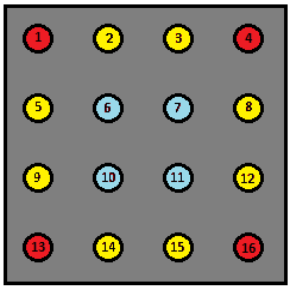


Figura 7

Los resultados obtenidos son los siguientes:

5 puntos: 154680

6 puntos: 1331944

7 puntos: 10690096

8 puntos: 79137824

9 puntos: 533427944

10 puntos: 3221413136

11 puntos: 17068504632

12 puntos: 77129797424

13 puntos: 285415667080

14 puntos: 811404606344

15 puntos: 1577602537520

16 puntos: 1577602537520

Todas estas combinaciones suman un total de 4350069806144 combinaciones. Este número de combinaciones lo tenemos que multiplicar por 9, que es el número de colores disponibles para cada combinación, obteniendo así un total de 39150628255296 combinaciones permitidas. Más de 39 trillones de combinaciones que la versión actual.

Con estas cifras podemos probar que el sistema propuesto es mucho más segura que la versión actual.

7. Evaluación de usabilidad con usuarios reales

Una vez terminada la aplicación, se ha buscado un grupo de voluntarios con distintos perfiles para poder tener una muestra representativa de los usuarios potenciales que podrían usar esta aplicación. Partiendo de que esta aplicación es una propuesta de mejora para el sistema de contraseña grafico de Android, suponemos que todos los voluntarios han usado previamente el sistema de contraseña actual y por lo tanto están familiarizados con su funcionamiento.

7.1 Resumen de la interacción con los voluntarios para las pruebas

Para realizar las pruebas no se les ha dado a los voluntarios ningún tipo de información sobre los problemas con el sistema actual o sugerencias de cómo se deberían de realizar las combinaciones. Intentando de esta manera que los resultados obtenidos puedan representar situaciones reales de uso.

El entorno general de uso para estas pruebas se basa en dejarle la aplicación al voluntario/a para que pueda juguetear con ella durante unos minutos. Con esto conseguimos que se familiarice con ella y pueda realizar varias pruebas antes de decidirse por el patrón que usara para desbloquear la pantalla.

Una vez que los voluntarios se han quedado satisfechos con el patrón seleccionado y lo ha probado unas suficientes veces para estar seguros que son capaces de recordarlo, se daría por terminado la primera fase de las pruebas.

En esta primera fase lo que se busca es poder obtener una idea de las preferencias de los voluntarios al elaborar su patrón.

En la segunda fase se pone a prueba el nivel de dificultad para recordar el patrón previamente seleccionado. Para conseguir esto, se le pedirá a los voluntarios aproximadamente 24 horas después de haber establecido su patrón, que vuelvan a intentar desbloquear la pantalla usando dicho patrón.

Para realizar esta tarea de una manera cómoda y fiable, se ha implementado un sistema de almacenamiento en ficheros dentro de la memoria interna, donde se guarda el hash del patrón que cada voluntario/a ha configurado. De esta manera, cada vez que los voluntarios van a usar por segunda vez la aplicación, simplemente con identificarse en la aplicación, se seleccionara automáticamente el último patrón que usaron en las pruebas del día anterior como el patrón actual. Esta funcionalidad es bastante útil si se utiliza el mismo teléfono para realizar las pruebas con distintos voluntarios.

También se almacena en un fichero todas las pruebas realizadas por los voluntarios tanto para establecer el patrón como para desbloquear la pantalla. A continuación se muestra un ejemplo de los datos obtenidos por el voluntario Pep.

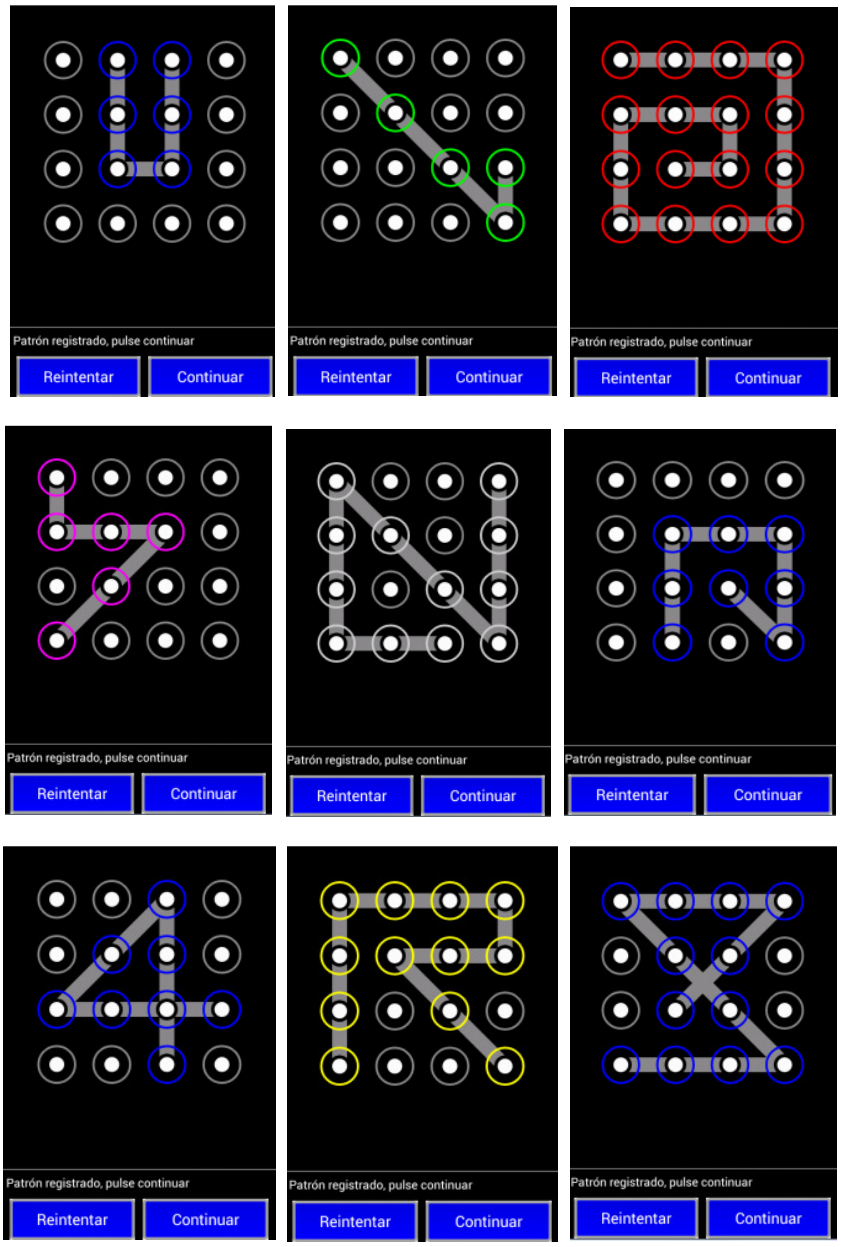
```
Pep registrando: Yellow 13 10 7 4 3 2 1 6 11 16     Sat Dec 07 20:49:21 2013
Pep confirmando: Yellow 13 10 7 4 3 2 1 6 11 16     Sat Dec 07 20:49:29 2013
## Patrón confirmado después de 1 intentos ##
Pep desbloquear: Yellow 13 10 7 4 3 2 1 6 11 16     Sat Dec 07 20:49:38 2013
## Pantalla desbloqueada después de 1 intentos ##
```

En este ejemplo se puede ver que se muestra el nombre del voluntario, el patrón seleccionado y la fecha cuando se registró. De esta manera podemos analizar fácilmente cual es el primer patrón que se selecciona, la dificultad para confírmalo antes de registrarlo y por último cuantos intentos son necesarios para conseguir desbloquear la pantalla.

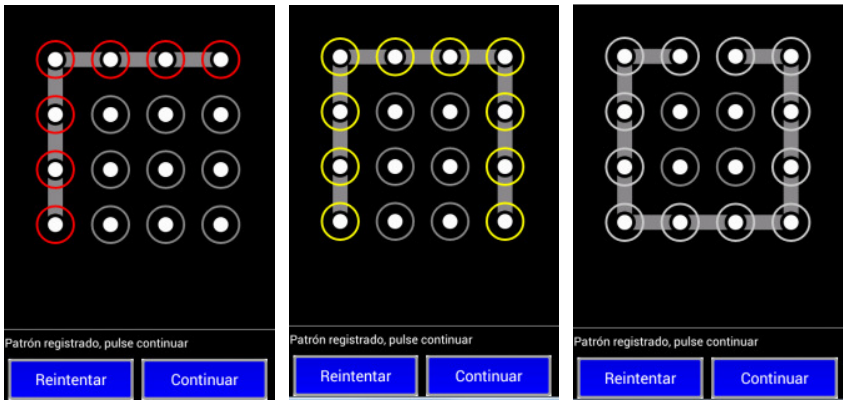
7.2 Análisis de los resultados obtenidos

Examinando los resultados de las pruebas se puede ver que más del 75% de los voluntarios usaron al menos dos de los cuatro puntos centrales de la cuadricula. En los casos donde no se usaron los puntos del medio, nos encontramos que al menos se utilizaron siete puntos. Menos un caso donde solo se usó el mínimo requerido, cinco puntos

A continuación se muestran algunos ejemplos de las combinaciones elegidas donde se han usado los puntos centrales.



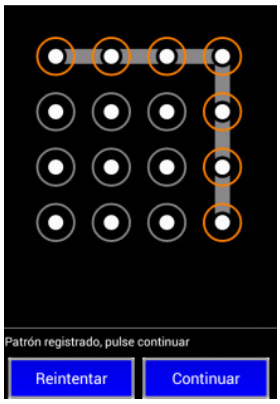
A continuación se muestran algunos ejemplos de las combinaciones elegidas donde no se han usado los puntos centrales.



Examinando en detalle las pruebas, podemos concluir que no suele haber problemas para recordar los colores. Sin embargo, cuando las secuencias que se intentaban eran muy largas, se encontraron con varios errores al confirmar dicho patrón. Como consecuencia se cambiaba el patrón por uno más sencillo.

En los casos en el que los voluntarios no tuvieron problemas confirmando el patrón seleccionado, casi no se encontraron fallos en el momento de desbloquear la pantalla. Los pocos fallos que se registraron, se podría decir casi con seguridad que fueron por problemas de selección del patrón en la pantalla y no por problemas de recordarlo correctamente. Esta deducción se obtiene al ver que siempre que se produjo un error fue porque se seleccionó un punto situado justo al lado del deseado y en el siguiente intento se consigue la secuencia correcta.

De las secuencias más cortas seleccionadas sin usar los puntos del medio, salvo una excepción, son todas de mínimo siete puntos. Donde curiosamente se ha registrado la única combinación que se ha repetido en las pruebas. Dicha secuencia se muestra a continuación. Como se puede ver el patrón elegido es relativamente sencillo y probablemente fácil de averiguar si se siguiesen los rastros de grasa dejados en la pantalla. Aun así, en este caso encontramos una mejora en relación con la versión actual donde solo se tendría que probar dicha secuencia 2 veces, para cubrir los 2 posibles sentidos y conseguir desbloquear la pantalla. Con el sistema propuesto, nos encontraríamos con 18 posibilidades, ya que tendrían que probar los 9 colores con las 2 posibles direcciones.



Si a este sistema se le añade un número reducido máximo de intentos para que se bloquee el terminal, se podría impedir que estos ataques por fuerza bruta tuviesen éxito en muchos de los casos.

También hay que mencionar, que solo uno de los voluntarios eligió un patrón donde se usaron todos los puntos. Que la secuencia más corta elegida fue de 5 puntos y que las secuencias que más se usaron fueron las de 6 y 12 puntos.

8. Conclusiones

Después de analizar el prototipo se puede concluir que el sistema propuesto ayudaría a mejorar la seguridad en este tipo de dispositivos, ya que en los casos donde los voluntarios eligieron un patrón con un color y ningún punto en el centro de la cuadricula, se obtuvo un número mínimo de siete puntos. Consiguiendo así un aumento en la seguridad de 2 a 18 posibilidades.

Los casos donde se eligieron al menos 2 puntos del centro de la cuadricula representan más del 75% de los resultados obtenidos. Esto se ha conseguido sin informar a los usuarios de la importancia de seleccionar los puntos centrales de la cuadricula para intentar camuflar los restos de grasa dejados en la pantalla con el uso normal de los terminales.

Con esto podemos concluir que si este sistema se usase con ciertas recomendaciones, se podría ver mejorada la seguridad sin que esto supusiese un problema al usuario para recordar su contraseña.

Bibliografía

JERMYN, I., MAYER, A., MONROSE, F., REITER, M., AND RUBIN, A. 1999. The design and analysis of graphical passwords. In Proceedings of the 8th USENIX Security Symposium.

KIM, D., DUNPHY, P., BRIGGS, P., HOOK, J., NICHOLSON, J., NICHOLSON, J., AND OLIVIER, P. 2010. Multi-touch authentication on tabletops. In Proceedings of the 28th ACMConference on Human Factors in Computing Systems (CHI). 1093–1102.

GAO, H., GUO, X., CHEN, X., WANG, L., AND LIU, X. 2008. Yagp: Yet another graphical password strategy. In Proceedings of the Annual Computer Security Applications Conference (ACSAC).

GOLDBERG, J., HAGMAN, J., AND SAZAWAL, V. 2002. Doodling our way to better authentication (student poster).In Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI).

WEISS, R. AND DE LUCA, A. 2008. PassShapes—utilizing stroke based authentication to increase password memorability. In Proceedings of the Nordic Conference on Human-Computer Interactions (NordiCHI).383–392.

TAO, H. AND ADAMS, C. 2008. Pass-Go: A proposal to improve the usability of graphical passwords. Int. J. Net. Secur. 7, 2, 273–292.

GRIDSURE. 2009. GrIDsure corporate website. http://www.gridsure.com.

PASSFACES CORPORATION. 2009. The science behind Passfaces. White paper. http://www.passfaces.com/ enterprise/resources/white_papers.htm.

DAVIS, D.,MONROSE, F., AND REITER, M. 2004. On user choice in graphical password schemes. In Proceedings of the 13th USENIX Security Symposium.

DHAMIJA, R. AND PERRIG, A. 2000. Déjà Vu: A user study using images for authentication. In Proceedings of the 9th USENIX Security Symposium.

BICAKCI, K., ATALAY, N. B., YUCEEL, M., GURBASLAR, H., AND ERDENIZ, B. 2009a. Towards usable solutions to graphical password hotspot problem. In Proceedings of the 33rd Annual IEEE International Computer Software and Applications Conference.

WIEDENBECK, S.,WATERS, J.,BIRGET, J.,BRODSKIY, A., ANDMEMON, N. 2005c. PassPoints: Design and longitudinal evaluation of a graphical password system. Int. J. Human Comput. Stud. 63, 1-2, 102–127.

BIRGET, J., HONG, D., AND MEMON, N. 2006. Graphical passwords based on robust discretization. IEEE Trans. Inf. Forensics Secur. 1, 3, 395–399.

CHIASSON, S., VAN OORSCHOT, P. C., AND BIDDLE, R. 2007b. Graphical password authentication using Cued Click Points. In Proceedings of the European Symposium on Research in Computer Security (ESORICS). Lecture Notes in Computer Science, vol. 4734, Springer, Berlin, 359–374.

CHIASSON, S., FORGET, A., BIDDLE, R., AND VAN OORSCHOT, P. C. 2008a. Influencing users towards better passwords: Persuasive Cued Click-Points. In Proceedings of the BCS Conference on Human Computer Interaction (HCI).