ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

Sistema de autenticación de dos factores basado en tarjeta inteligente y tecnologías NFC

(Two factor authentication using smart cards and NFC technologies)

Para acceder al Título de

Graduado en
Ingeniería de Tecnologías de Telecomunicación

Autor: Javier de Gregorio Menezo

Febrero - 2017

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Javier de Gregorio Menezo
Director del TFG: Jorge Lanza Calderón
Título: "Sistema de autenticación de dos factores basado en tarjeta inteligente y tecnologías NFC"
Title: "Two factor authentication using smart cards and NFC technologies"

Presentado a examen el día: 24 de febrero de 2017

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:

Presidente (Apellidos, Nombre): Casanueva López, Alicia

Secretario (Apellidos, Nombre): Sánchez González, Luis

Vocal (Apellidos, Nombre): García González, Alberto Eloy

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente	Fdo.: El Secretario
Fdo.: El Vocal	Fdo.: El Director del TFG (sólo si es distinto del Secretario)
Vº Bº del Subdirector	Trabajo Fin de Grado Nº (a asignar por Secretaría)

Agradecimientos

Me gustaría agradecer a mi director del trabajo de fin de grado, Jorge Lanza, por ofrecerme la posibilidad de trabajar en este proyecto y por guiarme durante estos meses en la realización del mismo. Así como a toda la gente del laboratorio de Telemática, que nunca dudan en echarte una mano.

A mis padres, por empujarme a seguir luchando, por el esfuerzo que han realizado y por toda la ayuda que me han brindado durante todos estos años. A mi hermana, por ser mi ejemplo a seguir. A Jose, por su total apoyo y generosidad.

A Verónica, por esperarme y estar siempre a mi lado.

Gracias.

Contenido

Índice de figuras

Índice de tablas

Resumen

En el actual mundo tecnológico, solventada la provisión de comunicación de forma ubicua, es cada vez más relevante poder garantizar la identidad de la persona que solicita el acceso a la información. Existen multitud de métodos que permiten llevar a cabo esta labor y de forma genérica para la identificación de un usuario se recurre a algo que "sabe" (número secreto, contraseña, PIN…), y/o algo que "es" (técnicas biométricas). Cada una de éstas técnicas por separado presenta ciertas deficiencias que pueden ser solventadas mediante su uso conjunto. Si además se logra incluir cierta variabilidad en los procedimientos que se implemente, el sistema será aún más robusto.

La combinación más comúnmente utilizada en la actualidad es la autenticación mediante lo que se tiene y lo que se conoce *(Two Factor Authentication, 2FA)*, mejorando notablemente la seguridad frente al tradicional empleo de duplas usuario-contraseña (lo que se conoce).

En este proyecto se estudiará el empleo de la tarjeta inteligente sola o en combinación con un terminal móvil como elemento habilitador de la autenticación de dos factores. Partiendo de los mecanismos estándares de generación de claves de un solo uso (*One Time Password, OTP*), se evaluará su habitual implementación en tarjetas inteligentes y se analizará la viabilidad de su uso en las situaciones de autenticación más habituales.

Abstract

In today's technological world, the target of providing communication anywhere almost being solved, it is increasingly important to be able to guarantee the identity of the user requesting access to information. There are plenty of methods to accomplish this work and the problem about identifying a user is determined by something you “know” (secret number, password, PIN...), and/or something that “is” (biometrics). Each of these techniques separately used, introduce some vulnerabilities that could be solved by joining the use of both of them. Additionally, if you manage to include some sort of variability in the procedures implemented, the system will be even more secure.

Nowadays, the most commonly used combination is the two-factor authentication (2FA) (what you have and what is known), improving the traditional use of user-password pairs (what is known).

In the present project, the use of a smart card will be studied, either way in single-use or in combination with a mobile terminal as enabler of the two-factor authentication. Starting from the standard key generation mechanisms for single use (*One Time Password*, *OTP*), its usual implementation on smart cards will be defined and its feasibility use in common authentication situations will be analyzed.

Acrónimos

APDU	*Application Protocol Data Unit*
API	*Application Programming Interface*
ATR	*Answer To Reset*
ATS	*Answer To Select*
CID	*Card Identifier*
CPU	*Central Processing Unit*
EEPROM	*Electrically Erasable Read Only Memory*
FSDI	*Frame Size Device Integer*
HMAC	*Hash-based Message Authentication Code*
HOTP	*HMAC-based One Time Password*
HTTP	*Hypertext Transfer Protocol*
IEC	*International Electrotechnical Commission*
ISM	*Industrial, Scientific and Medical*
ISO	*International Organization for Standardization*
JCRE	*Java Card Runtime Environment*
JRE	*Java Runtime Environment*
JSON	*JavaScript Object Notation*
NDEF	*NFC Data Exchange Format*
NFC	*Near Field Communication*
OCR	*Optical Character Recognition*
OTP	*One Time Password*
PCD	*Proximity Coupling Device*
PC/SC	*Personal Computer/Smart Card*
PIN	*Personal Identification Number*
QR	*Quick Response*
RAM	*Random Access Memory*
RATS	*Request for Answer To Select*
REST	*Representational State Transfer*
RFC	*Request For Comments*
RFID	*Radio Frequency Identification*
RFU	*Reserved for Future Use*
ROM	*Read Only Memory*
RTD	*Record Type Definition*

SHA	*Secure Hash Algorithm*
SIM	*Subscriber Identity Module*
SMS	*Short Message Service*
SOAP	*Simple Object Access Protocol*
SPU	*Standard or Proprietary Use*
SSL	*Secure Sockets Layer*
TCP	*Transmission Control Protocol*
TOTP	*Time-based One Time Password*
TPDU	*Transmission Protocol Data Unit*
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
USB	*Universal Serial Bus*
VCD	*Vicinity Coupling Device*
VPN	*Virtual Private Network*
WSDL	*Web Services Description Language*
XML	*eXtensible Markup Language*

1 Introducción y objetivos

1.1 Introducción

Hoy en día, una de las principales amenazas de los sistemas de computación es el ataque a servidores y bases de datos, ya que éstas representan el núcleo de cualquier organización y almacenan información de clientes y otros datos confidenciales. En este ámbito, la técnica más utilizada es el *phishing*, que consiste en engañar al usuario para robarle información confidencial, generalmente claves de acceso. Esto se consigue mediante ingeniería social; se contacta con el usuario aparentando ser una persona conocida o un sitio de confianza y se le solicita información. Hasta ahora, estos ataques se realizaban mediante correo electrónico, pero el uso masivo de redes sociales, banca *online* y dispositivos móviles con conexión a internet, ha multiplicado este tipo de ataques.

Ante las debilidades mostradas, es necesaria la implementación de metodologías adicionales que refuercen la seguridad en la identificación de usuarios. El principal objetivo de una solución de autenticación es asegurar el acceso controlado a información sensible, y para ello hay que comprobar, de forma fehaciente, que aquel que accede es quien dice ser. En la actualidad, existen multitud de métodos que permiten llevar a cabo dicha labor [1]. De forma genérica, para identificar a un usuario se recurre a algo que sabe (número secreto, contraseña, etc.), algo que porta o tiene (tarjeta identificadora, dispositivo, etc.) y/o algo que es (técnicas biométricas). Cada una de estas técnicas, por separado, presenta deficiencias que pueden ser solventadas mediante su uso conjunto, resultando en procedimientos de autenticación de dos y tres factores.

Tal y como se ha introducido, el método más habitual de identificación basado en el empleo de, únicamente, la dupla usuario/contraseña, evidencia grandes deficiencias como demuestran la multitud de ataques, fraudes y reemplazos de identidad que se están llevando a cabo en los últimos tiempos. Algunos ejemplos de estas debilidades y su posible explotación por parte de usuarios malintencionados son el robo masivo de cuentas a Dropbox en 2012 [2] o la filtración de fotografías de celebridades en el ataque a la plataforma de almacenamiento de Apple, iCloud, en el año 2014 [3].

Para tratar de solventar la problemática detectada en la identificación, se introduce el empleo de elementos biométricos como elemento adicional a aquello que el usuario conoce. La identificación mediante biometría reconoce al usuario basándose en quien es, fijándose en una

única e inalterable característica humana que no puede ser perdida, olvidada, sustraída o duplicada. Sin embargo, a pesar de que el uso del dispositivo biométrico es capaz de identificar de manera inequívoca al usuario (considerando la no existencia de error en la adquisición de la información), la variabilidad de los datos que se emplean en el procedimiento de autenticación es nula. Así, el modelado de la respuesta ante el lector biométrico empleado puede garantizar el acceso, si bien el usuario real no ha sido quien lo ha solicitado, como es el caso de la vulnerabilidad demostrada en los teléfonos móviles desbloqueables mediante huella dactilar [4]. Algunas soluciones, como los dispositivos que realizan prueba de vida en las técnicas biométricas, son posibles, pero incrementan notablemente el coste. Adicionalmente, es importante reseñar que algunas de las características biométricas de una persona tienden a variar con el tiempo, lo que redundará en falsos negativos a la hora de identificar al usuario.

De lo anterior, se extrae la necesidad de buscar una solución adicional que permita incrementar la variabilidad en el procedimiento de identificación e incluso para garantizar aún más la seguridad que la información que almacene y los procedimientos para generar el dato identificativo sean en parte o totalmente desconocidos para el usuario. Actualmente, la combinación más utilizada es la autenticación de dos factores, mediante lo que se conoce y lo que se posee. En el más común de los casos, suele tratarse de un código adicional que se envía al correo electrónico o a un teléfono móvil mediante mensaje SMS.

La evolución de las nuevas tecnologías ha obligado a tratar de realizar una integración de los diferentes protocolos de comunicación en soluciones de interoperabilidad basadas en estándares. Por ello, la organización IETF (*Internet Engineering Task Force*) es la encargada de de regular las propuestas y los estándares, conocidos como RFC (*Request For Comments*). En este contexto, se encuentra el RFC 2289 [5], donde se describe la autenticación basada en contraseñas de un solo uso (*One Time Password, OTP*) para acceder a un sistema y prevenir el problema de los ataques basados en la captura de contraseñas reutilizables.

Los sistemas de autenticación basados en contraseñas de un solo uso están diseñados para hacer frente al tipo de vulnerabilidades mencionadas, como robo de cuentas o suplantación de identidad. Normalmente, el código OTP es utilizado como un mecanismo adicional, es decir, hay un primer paso de autenticación regular basada en la contraseña habitual y un segundo paso que requiere el empleo del código OTP. Esto es lo que se conoce como autenticación de dos factores. Generalmente, una contraseña de un solo uso es una cifra de entre seis y ocho dígitos con una validez temporal limitada que es generada por algún dispositivo que debe conocerse para acceder a un sistema.

Hasta la aparición de los teléfonos móviles, para responder a las necesidades de movilidad, muchas compañías utilizaban unos pequeños dispositivos, como el mostrado en la

imagen de la Figura 1, para garantizar la seguridad del acceso remoto a sus VPN (*Virtual Private Network*). Éstos eran unos dispositivos con un botón que, cuando era pulsado, suministraban el código OTP con el que se debía autenticar el acceso del usuario.

Figura 1. Token de seguridad SecurID de RSA

La expansión de los teléfonos móviles inteligentes ha permitido que estas soluciones se trasladen hacia el mundo de las aplicaciones móviles, lo que provoca, de nuevo, los mismos problemas de seguridad que el acceso desde ordenadores, ya que son vulnerables a virus informáticos y ataques. El número de ataques maliciosos a dispositivos móviles se incrementa cada año. Así, un estudio de *GData* detectó en 2014 una cantidad de 1,5 millones de programas maliciosos para dispositivos Android [6]. Por ello, es necesaria la utilización de un elemento/procedimiento externo, independiente y seguro para la obtención del código OTP.

Las tarjetas inteligentes son posiblemente los dispositivos más apropiados para esta tarea, debido a su robustez y seguridad, además de otras características favorables como su tamaño, portabilidad, coste y capacidad. Las tarjetas inteligentes incorporan funcionalidades que las hacen especialmente útiles en la generación y custodia de claves secretas o la ejecución de los algoritmos criptográficos involucrados. Todo ello unido a su alto grado de penetración en la sociedad actual y las perspectivas de crecimiento en los próximos años, con una tasa de crecimiento anual media (*Compound Annual Growth Rate, CAGR*) del 8.4% [7] en el periodo entre 2016 y 2024, las convierten en un elemento atractivo a emplear en la identificación.

1.2 Motivación y objetivos

La principal motivación para la realización de este proyecto es evaluar la viabilidad de emplear una tarjeta inteligente como elemento generador de las contraseñas de un solo uso. Para dotar de una mayor usabilidad a la solución y dado el alto grado de inclusión de la telefonía móvil, se pretende emplear dichos dispositivos en combinación con la propia tarjeta inteligente y el uso de las tecnologías NFC para hacer más accesible el sistema planteado. La implementación práctica de un sistema de este tipo permite la adquisición de los conocimientos básicos necesarios para poner en marcha una arquitectura de control de acceso a la información.

Por ello, el objetivo de este proyecto es desarrollar un sistema real que permita, principalmente, reforzar la seguridad en el proceso de autenticación de un usuario empleando soluciones basadas en OTP y tratar de resolver los diferentes problemas de vulnerabilidad que conllevan las contraseñas estáticas habituales. Generalmente, una contraseña estática ofrece una seguridad débil, por lo que, aprovechando que actualmente cualquier usuario dispone de un teléfono móvil, se puede crear un sistema más robusto, añadiendo una nueva capa de seguridad y de manera sencilla para el usuario.

En este proyecto, se empleará una tarjeta inteligente en combinación con un terminal móvil como elemento habilitador. Será la tarjeta la encargada de la generación del código OTP de forma que los algoritmos y claves se mantengan almacenados de forma robusta. Adicionalmente, se experimentará con la viabilidad de evitar la interacción del usuario con el propio código OTP, haciendo más amigable el proceso.

En resumen, los objetivos del proyecto son los siguientes:

- Conocimiento de los algoritmos básicos de generación de palabras clave de un solo uso, tanto desde el punto de vista teórico (base criptográfica, etc.), como desde el operativo.
- Aprendizaje de las características básicas de las tecnologías de tarjeta inteligente y NFC, logrando identificar las sinergias entre ambas y explotarlas en el diseño e implementación de una solución de seguridad.
- Desarrollo de una solución que permita ofrecer un mecanismo de autenticación robusto para la identificación en cualquier tipo de aplicación, sustentándose en la integración e interacción de muy diversas tecnologías de comunicaciones, almacenamiento, etc.

1.3 Organización del documento

Para tratar de abordar con éxito los objetivos planteados, a continuación, se describe la organización de este documento, cuya estructura se ha dividido en cuatro capítulos. A lo largo del primer capítulo**,** tras una breve introducción, se ha situado el entorno tecnológico en el que el contexto del trabajo se va a desarrollar.

En el **capítulo 2**, se introduce la autenticación en el acceso a sistemas y se tratan los conceptos teóricos relacionados con las herramientas utilizadas para este proyecto. Se describen los sistemas de autenticación en dos factores, los algoritmos TOTP y HOTP empleados, la arquitectura y características de las tarjetas inteligentes con y sin contacto, y el uso de las tecnologías NFC.

Seguidamente, en el **capítulo 3**, se describe el desarrollo del sistema de autenticación de dos factores propuesto para este proyecto y se detalla el diseño y la metodología de cada módulo que compone dicho sistema.

Por último, en el **capítulo 4**, se recopilan las principales conclusiones obtenidas y se proponen nuevas líneas de futuro surgidas de las posibles vulnerabilidades que presenta el sistema implementado.

2 Conceptos teóricos

La autenticación en sistemas ha alcanzado un notable desarrollo en las últimas décadas debido a la multitud de servicios de internet que requieren la identificación de los usuarios, especialmente para el acceso a los recursos y datos personales. A continuación, se procede a describir los sistemas de autenticación, así como los diferentes conceptos teóricos y protocolos de generación de claves de un solo uso relacionados con la autenticación de dos factores.

2.1 Autenticación

La autenticación es el proceso de confirmación que se realiza a través de medios electrónicos de la identidad de un individuo o de un organismo para acceder a determinados recursos o realizar determinadas tareas. Todo usuario que necesite acceder a un sistema requiere de un proceso de identificación y autenticación, para comprobar que la persona es quien dice ser.

Los diferentes métodos de autenticación se pueden dividir en tres categorías en función del elemento que se utilice para la verificación de la identidad del usuario:

i. Sistemas basados en **algo que conoce**: código PIN (*Personal Identification Number*), contraseñas, etc. El modelo de autenticación más básico consiste en decidir si un usuario es quien dice ser simplemente basándonos en una prueba de conocimiento que *a priori* sólo ese usuario puede superar.
ii. Sistemas basados en **algo que posee**: tarjetas inteligentes, tokens de seguridad, claves de banco, etc. En este tipo de sistemas, el usuario emplea un dispositivo externo para realizar su identificación.
iii. Sistemas basados en una **característica física del usuario**: biometría. Sistemas que identifican al usuario fijándose en una única e inalterable característica humana que no puede ser perdida, olvidada, sustraída o duplicada.

2.2 Autenticación de dos factores

Un sistema de autenticación de dos factores (*Two Factor Authentication*, *2FA*), también conocida como verificación en dos pasos (*Two-Step Verification*, 2SV), implica que el usuario necesita verificar su identidad a través de dos canales seguros e independientes antes de poder iniciar sesión o utilizar un servicio. De esta forma, en el caso de que uno de los canales se vea comprometido, la necesidad de finalizar el proceso con la información del otro canal evita accesos no deseados o no autorizados. Como se muestra en la Figura 2, un canal emplearía el

método de autenticación habitual de contraseña estática y el otro canal, el suministro del código OTP, mediante llamada de voz, mensaje SMS, correo electrónico, etc. dependiendo del tipo de sistema que se utilice.

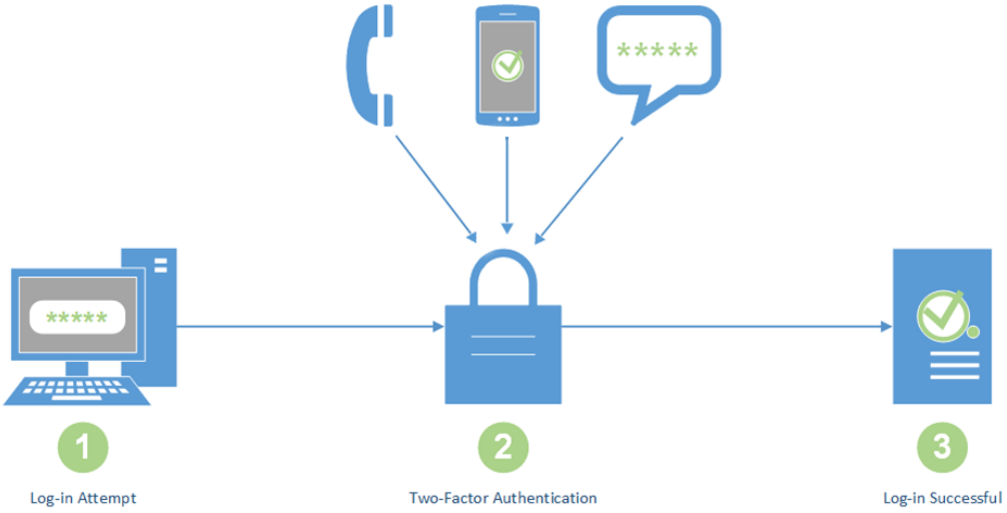

Figura 2. Proceso de autenticación de dos factores.

Usualmente, el primer canal consiste en información proporcionada por el usuario tal como una contraseña, en principio conocida únicamente por él mismo. El otro canal incluye información no generada por el usuario, generalmente provista por el propio servidor en el que se quiere autenticar o en colaboración con él. Dicha información incluye una elevada variabilidad y cambia cada vez que se desea realizar la autenticación. Adicionalmente, la validez de esa información tiene una limitación temporal definida y suele ser de un único uso, lo que evita la repetición de mensajes y por tanto refuerza la seguridad.

Los códigos OTP generados para ese segundo canal suelen hacer uso de aleatoriedad o pseudoaleatoriedad, siguiendo procedimientos criptográficos, lo que dificulta a un posible atacante la predicción de sucesivos códigos. A continuación, se describen algunas de las diferentes soluciones que se emplean en función de cómo el código es generado.

2.2.1 Por eventos

Este tipo de dispositivos genera un código de un solo uso a partir de un evento, ya sea pulsando un botón o acercando el dispositivo a un lector. Las implementaciones más habituales, que se muestran en la imagen de la Figura 3, son el uso de tarjetas inteligentes, tokens de generación de códigos y YubiKeys [8]. Los tokens de seguridad, como ya se ha mencionado en la introducción, son unos dispositivos de seguridad adicional con un botón que, cuando es pulsado, generan el código OTP con el que se debe autenticar.

Figura 3. Tarjeta inteligente, 2FA token y Yubikey.

Las **tarjetas inteligentes** requieren de un lector especial para el intercambio de información, ya sea a través de sus contactos metálicos o a través de su antena si es una tarjeta sin contacto o de proximidad. Contienen un chip que actúa como un pequeño ordenador independiente con su propio procesador, memoria y capacidad criptográfica.

Los **2FA tokens** también disponen de un procesador independiente, pero en este caso, no se conectan a ningún otro dispositivo. En su lugar, tienen una pequeña pantalla que muestra un nuevo código cada vez que un botón es pulsado.

Los **YubiKeys** también tienen su propia capacidad criptográfica, pero se comunican a través de una conexión USB a un dispositivo simulando ser un teclado. Cada vez que se conecta el YubiKey, genera un nuevo código y lo escribe de manera automática.

2.2.2 Desde el servidor

Actualmente, el método más extendido, es el uso de los **mensajes de texto** (*SMS, Short Message Service)* o **correo electrónico** para la recepción del código OTP. Tras la autenticación con nombre de usuario y contraseña habitual, se pide al usuario que ingrese un código numérico adicional. Dicho código se envía desde el servidor a su dispositivo móvil mediante un mensaje de texto, como se muestra en los ejemplos de la imagen de la Figura 4, o mediante correo electrónico. Cada mensaje solicitado envía un código diferente.

El canal empleado para recibir el código OTP es un canal que requiere conexión, además de pertenecer a una red de operador móvil. Todo ello hace que surjan algunos inconvenientes, como son, por ejemplo, la existencia de problemas para que un proveedor de telefonía no pueda entregar el SMS debido a congestión, restricciones regionales para recibir mensajes o que el operador en cuestión no tenga cobertura en el punto donde el usuario se encuentra.

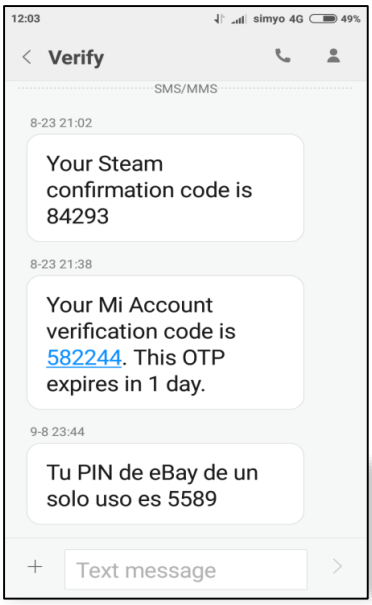

Figura 4. Ejemplos de códigos OTP enviados a través de SMS.

Una solución similar consiste en el empleo del canal de voz en lugar del mensaje de texto. Por ejemplo, como se observa en la Figura 5, cuando se procede a verificar un número de teléfono en la aplicación de WhatsApp [9], se ofrece la posibilidad de recibir una llamada de voz donde se comunicará el código OTP. De esta forma, se extiende la operativa para casos donde no se disponga de servicio SMS o falle dicho servicio.

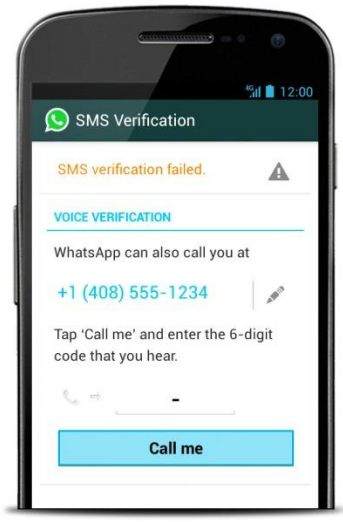

Figura 5. Ejemplo de envío de código OTP mediante canal de voz.

Otro método de autenticación, aunque actualmente esté en desuso, es el sistema de **TANs *(Transaction Authentication Number)*** y se utilizaba principalmente en los bancos. Se enviaba al usuario una lista de códigos generados previamente por el banco, de manera que, en función de la operación que se quería realizar, el usuario debía ingresar el código correspondiente.

2.2.3 De forma local

Este sistema realiza un tipo de servicio similar al sistema mediante mensajes SMS, pero, en lugar de enviar el código, se genera localmente en el propio dispositivo a partir de una información que le haya enviado el servidor en un primer momento. Con la actual extensión masiva de los teléfonos móviles inteligentes, la **aplicación móvil** se ha convertido en protagonista. Este tipo de autenticación se basa en los algoritmos criptográficos para contraseñas de un solo uso basadas en tiempo (*Time-based One Time Password*, TOTP) que se han mencionado en el capítulo de introducción.

La operativa básica consiste en la creación de una clave secreta (semilla) que es generada por el servidor. En un primer momento, el servidor comparte este secreto, por ejemplo, mediante la lectura de un código QR (*Quick Response*), de manera que la aplicación almacene la misma semilla con la que generar los mismos códigos de autenticación. En la Tabla 1, se muestra el ejemplo de un código QR y su contenido. Este secreto se codifica de forma criptográfica en combinación con la hora y fecha actuales generando un código de validez temporal, limitado generalmente a 30 ó 60 segundos.

Tabla 1. Código QR y su contenido.

	`otpauth://totp/Google%3Auser_email%40gmail.com?secret=12345678901234567890123456789012&issuer=Google`

Siempre que el tiempo o reloj del sistema local esté en sincronía con el del servidor y la semilla se haya importado correctamente, la aplicación generará códigos de inicio de sesión que coincidirán con los calculados en el servidor. Mientras que el secreto compartido no sea obtenido por un tercero, resultará muy difícil reconstruir la secuencia de códigos generados.

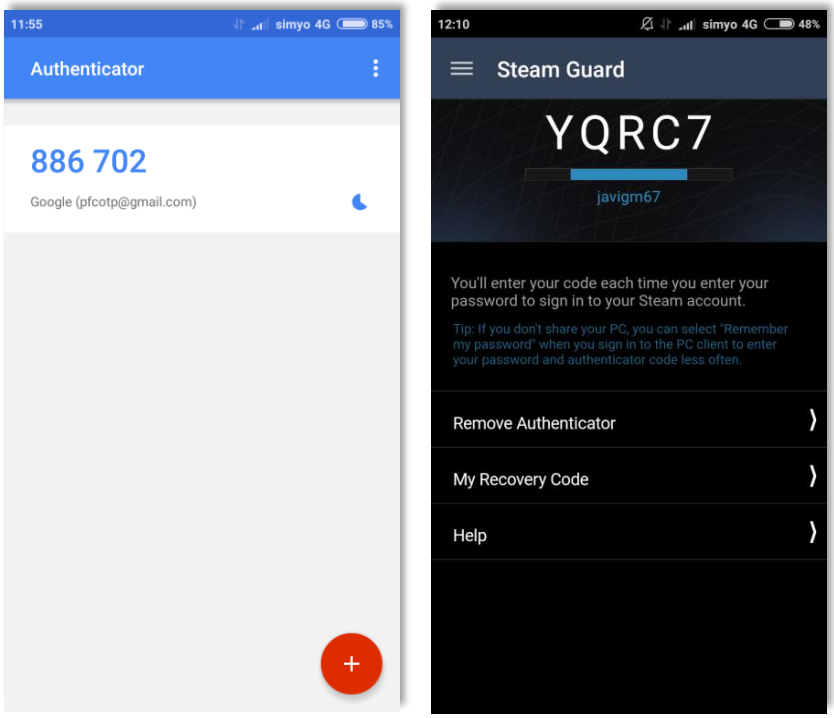

Figura 6. Ejemplos de códigos TOTP para Android; Google Authenticator y Steam Guard.

Existen varias implementaciones para este tipo de servicio, pero Google Authenticator probablemente sea la más conocida. En la Figura 6 se muestran dos ejemplos de capturas de pantalla de las aplicaciones de Google y Valve, donde los códigos TOTP que aparecen en pantalla, van cambiando cada 30 segundos.

2.3 Algoritmos One Time Password (OTP)

Los algoritmos de generación de los códigos OTP suelen hacer uso de aleatoriedad o pseudoaleatoriedad, lo que dificulta a un posible atacante la predicción de sucesivos OTPs. Además, se utilizan funciones de resumen (*hash*), cuyo resultado es difícil de revertir, esto es, de obtener la información que fue utilizada para crear el *hash*. Gracias a esta característica resulta prácticamente imposible predecir la sucesión de los códigos OTP observando los anteriores. A continuación, se describen los algoritmos para la creación de un código OTP.

2.3.1 HMAC-based One Time Password (HOTP)

El sistema HOTP permite la generación de contraseñas de un solo uso basadas en el mecanismo criptográfico HMAC (*Hash-based Message Authentication Code*) [Anexo I] descrito en el RFC 2104 [10]. Este algoritmo está basado en el valor de un contador que se va incrementando (*factor móvil*) y en una llave simétrica y estática (*clave secreta*) conocida únicamente por el dispositivo de autenticación y el servicio de validación. En la Figura 7, se muestran los pasos del proceso de generación del código OTP.

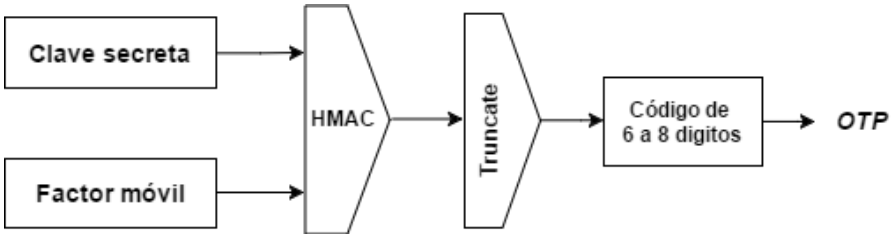

Figura 7. Proceso general de obtención del código OTP.

Para describir el procedimiento de obtención del valor HOTP siguiendo el RFC 4226 [11], suponemos el empleo del algoritmo *HMAC-SHA-1*, donde su salida es de 20 bytes (160 bits) y éstos deben ser truncados a algo que pueda ser introducido fácilmente por un usuario:

$$HOTP(K, C) = Truncate\,(HMAC\_SHA\_1\,(K, C))$$

La clave *K*, el contador *C* y los valores de datos, son resumidos (*hashed*) comenzando primero por los bytes de orden alto.

La operación *Truncate* representa la función que convierte el valor *HMAC-SHA-1(K, C)* en un HOTP. Esta operación de truncamiento se lleva a cabo en tres pasos:

a) Se genera un valor *HS = HMAC-SHA-1(K, C)*. El valor HS es una cadena de 20 bytes.
b) Se genera una secuencia de 4 bits (truncamiento dinámico):
    - Se toman los 4 últimos bits del último byte de HS (HS[19]), que será el offset del byte a partir del cual se toman los 4 bytes quitándole el bit más significativo (para evitar problemas con el signo), es decir, 31 bits.
c) Se calcula el valor del HOTP:
    - Se convierten esos 4 bytes (menos 1 bit) en un número, N, entre 0 y 2^{31}-1.
    - El HOTP se obtiene tomando el módulo $N\%10^{\{dígitos\}}$ y, por tanto, resultando un número entre *0* y $10^{\{dígitos\}}$*-1*.

Por ejemplo, supongamos que el resultado del *HMAC-SHA-1* de 20 bytes de longitud de la Tabla 2 ha sido el siguiente:

Tabla 2. Ejemplo de un valor de HMAC-SHA-1.

nº byte	00	01	02	03	04	05	06	07	08	09	**10**	11	12	13	14	15	16	17	18	19
valor	1F	86	98	69	0E	02	CA	16	61	85	**50**	**EF**	**7F**	**19**	DA	8E	94	5B	55	5A

El último byte vale 0x5A, el valor decimal de sus 4 últimos bits es 10 (0x0A). Por tanto, éste es el *offset* a partir del cual se toman los siguientes 31 bits. Es decir, 0x**50EF7F19**. En consecuencia, si queremos un HOTP de 6 dígitos:

HOTP = (0x50EF7F19) módulo 10^6 = 872921.

Las implementaciones del HOTP deben obtener un resultado de 6 dígitos como mínimo, con posibilidad de obtener códigos de 7 y 8 dígitos en función de la seguridad requerida.

2.3.2 Time-based One Time Password (TOTP)

Un TOTP (*Time-based One Time Password*) es un código de un solo uso, cuyo periodo de validez está limitado. Se trata de una metodología de generación de OTP en la que el cliente y el servicio de autenticación deben estar sincronizados, es decir, los relojes de ambos sistemas no deben tener un desajuste superior a un valor previamente definido. Además, a diferencia de HOTP, no tiene por qué existir conexión entre el autenticador y solicitante de autenticación, más allá de la comunicación obligatoria existente durante la inicialización del sistema.

La organización IETF ha aprobado un algoritmo para generar un TOTP estandarizado [12], basado en el HOTP (*HMAC-based One-Time Password Algorithm*) [11], que a su vez es una extensión del OTP [5]. En el caso del TOTP, el factor móvil se reemplaza el contador por un valor *T*, obtenido a partir de un tiempo de referencia y un intervalo (*time step*). Es decir, TOTP está definido como:

$$TOTP(K,T) = HOTP(K,C)$$

donde a su vez, *T* está definido como:

$$T = \frac{Tiempo\_Actual - T_0}{X}$$

donde el tiempo de referencia T_0 es, por defecto, el Unix *epoch*, es decir, 0. El tiempo Unix actual se define como el número de segundos transcurridos desde el *epoch*, esto es, desde el 1 de enero de 1970. El intervalo temporal (*time step*) *X* es de 30 segundos por defecto. Para calcular *T* se emplea la división entera, por lo que si, por ejemplo, T_0=0, *X*=30 y el tiempo actual fuese 59 segundos, sería *T*=1. En cambio, si el tiempo actual fuese 60 segundos, la división temporal da lugar a un valor *T*=2.

2.4 Tarjeta inteligente

Una tarjeta inteligente (*smart card*) es un circuito integrado enmarcado en un soporte plástico para su portabilidad y manejo [13]. Las tarjetas inteligentes tienen la capacidad de proteger y procesar datos almacenados en ellas. Estas características hacen de la tarjeta inteligente uno de los dispositivos más adecuados para la generación y custodia de claves

secretas, ejecución de algoritmos criptográficos, o por extensión el elemento idóneo para almacenamiento y procesado de información confidencial.

Es frecuente denominar tarjetas inteligentes a todas las tarjetas que poseen contactos dorados o plateados sobre su superficie. Sin embargo, dependiendo de las características del chip se pueden clasificar, según se refleja en la Figura 8, en tarjetas de memoria y tarjetas inteligentes. Las tarjetas de memoria disponen de un chip que únicamente ofrece capacidades como almacén de datos y pueden ser de acceso libre o de acceso restringido, dependiendo de la seguridad que requiera el contenido que guardan.

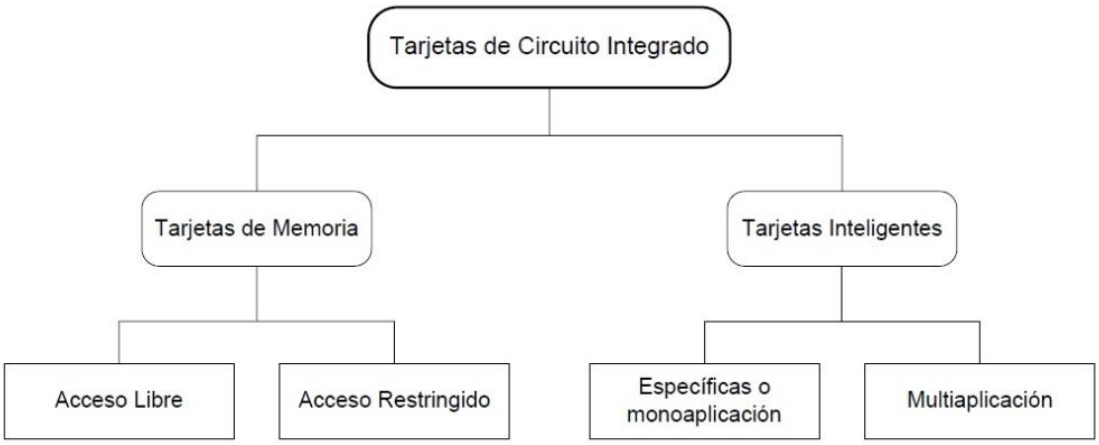

Figura 8. Clasificación de las tarjetas.

La principal característica que diferencia una tarjeta de memoria de una inteligente, es la introducción de un microprocesador en su circuito integrado. Este elemento es el que las hace inteligentes, permitiendo ejecutar comandos y trabajar con los datos que contiene. Las tarjetas con microprocesador son bastante flexibles puesto que pueden realizar muy diversas funciones. En el caso más simple sólo contienen datos referentes a una aplicación específica. No obstante, los sistemas operativos de las tarjetas más modernas hacen posible que puedan integrar varios programas sobre un único soporte.

Las tarjetas inteligentes transfieren sus datos a través de los contactos de su superficie, el deterioro de estos contactos es, en base a la experiencia, el mayor causante de fallo. Como evolución y mejora, se desarrolla la tarjeta inteligente sin contacto, utilizando campos magnéticos para la transferencia de datos. Sin embargo, a pesar de evitar el deterioro, presentan otros inconvenientes como la potencia necesaria para su activación o su mayor coste de fabricación.

2.4.1 Arquitectura

Las tarjetas inteligentes adoptan, en su mayoría, una arquitectura de maquina *Von Neumann*. El microcontrolador de una tarjeta inteligente está compuesto por un microprocesador y tres tipos de memoria, como se muestra en la Figura 9: ROM (*Read Only Memory*), EEPROM (*Electrically Erasable Read Only Memory*) y RAM (*Random Access Memory*). El microprocesador y la memoria están fabricados sobre el mismo chip, haciendo caro y difícil interceptar las señales que se intercambian entre ambos, proporcionando así una alta seguridad física de los datos almacenados en memoria.

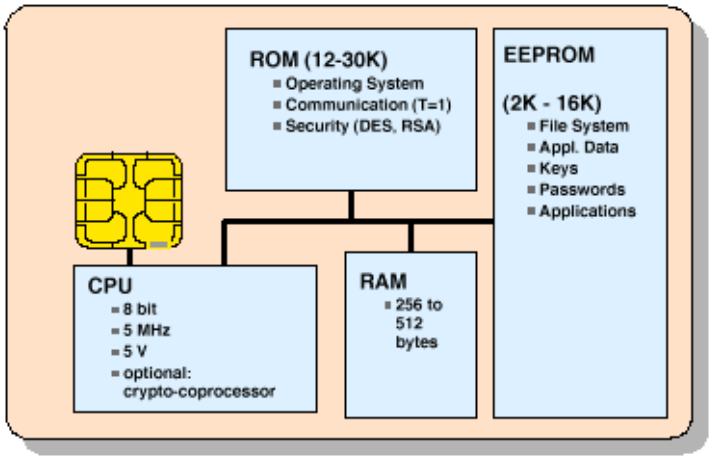

Figura 9. Arquitectura del circuito integrado de una tarjeta inteligente

- La unidad central de proceso o CPU (*Central Processing Unit*) es la encargada de realizar los cálculos y procesar la información.
- La memoria ROM se utiliza para almacenar los datos permanentes escritos en la fase de fabricación de la tarjeta: sistema operativo, aplicaciones y datos de usuario fijos. No necesita alimentación para mantener este tipo de memoria.
- La memoria EEPROM es una memoria no volátil que puede ser programada y borrada eléctricamente. En ella se encuentran los datos de usuario y aplicación, así como información bajo control del sistema operativo.
- La memoria RAM es la memoria de trabajo del sistema operativo y se utiliza para mantener los datos temporales de una sesión. Es volátil, por lo que la información se pierde al desconectar la alimentación. En lo referente a la lectura es igual de rápida que una EEPROM, pero en cuanto en la escritura es mucho más rápida.

De las tres memorias, la memoria ROM es la más barata, pues ocupa menos espacio por celda. Una celda de EEPROM ocupa hasta cuatro veces más que una de ROM, y una de RAM, aproximadamente, cuatro veces una de EEPROM. Ésta es la razón por la que las tarjetas

contienen pequeñas cantidades de RAM. En el mercado, encontramos chips que poseen desde 4K a 256K de ROM, de 1K a 256K de EEPROM y de 256 bits a 10K de RAM. La capacidad de memoria de una tarjeta inteligente está claramente limitada, por lo que se usan algoritmos de compresión de datos para alojar la información en su interior.

2.4.2 Características físicas

Por razones de compatibilidad, la forma y el tamaño de las tarjetas inteligentes están estandarizados, los formatos más utilizados son el ID-1 (por ejemplo, las tarjetas de crédito) y el ID-000 (tarjetas para telefonía móvil). Actualmente, además de estos formatos, existen otros como consecuencia de la evolución de los teléfonos móviles:

- El formato **ID-1** o **1FF**, para tarjetas SIM (*Subscriber Identity Module*), con unas dimensiones de 85.60×53.98 mm, es el más utilizado. Se puede ver este formato en las tarjetas bancarias, documentos de identidad, de transporte público, etc.
- El formato **ID-000** o **2FF** (Mini SIM), con unas dimensiones de 25×15 mm, surge como requerimiento para dispositivos tecnológicos pequeños, como es la telefonía móvil, que requieren el uso de una tarjeta de tamaño reducido.
- Los formatos **3FF** (Micro SIM) y **4FF** (Nano SIM) son los más reducidos, de 15x12 mm y 12.3x8.8 mm respectivamente, aunque mantienen el tamaño de los contactos del chip.

La presencia de los contactos del chip en su superficie, es otra de las características físicas de las tarjetas inteligentes. Sirve de enlace entre el dispositivo lector y la tarjeta. Es la vía por donde el microprocesador toma la alimentación para sus circuitos y lleva a cabo la transmisión de datos (Figura 10). La disposición, tamaño y contenido de los contactos está definida en el ISO 7816-2 [14].

Figura 10. Contactos del chip de una tarjeta inteligente

- **Vcc** se utiliza para suministrar alimentación. El voltaje que se aplica es de 3V ó 5V.
- **RST**, señal de *reset*, utilizado para resetear las comunicaciones de la tarjeta.
- **CLK** proporciona una señal de reloj externa que se tomará como referencia para la señal del reloj interno.
- **GND** es la conexión a masa.

- **Vpp** es un conector en desuso, aunque se conserva por motivos de compatibilidad con el estándar ISO 7816. Actualmente está designado como SPU (*Standard or Proprietary Use*) para uso de entrada o salida.
- **I/O** se utiliza para transferir datos entre el dispositivo lector y la tarjeta en modo *half-duplex*. El puerto de entrada/salida consiste en un registro a través del cual la información es transferida bit a bit.
- **C4** y **C8** son conectores RFU (*Reserved for Future Use*).

2.4.3 Protocolo de comunicación

Toda comunicación con una tarjeta inteligente siempre es iniciada por un dispositivo externo. Una tarjeta nunca transmite información sin que se haya producido antes una petición externa, lo que supone una relación maestro-esclavo, siendo el dispositivo el maestro y la tarjeta, el esclavo.

La transmisión de datos se realiza empleando un modelo asíncrono. Debido a la existencia de un único canal entre los dispositivos externos y la tarjeta, ambos deben turnarse para llevar a cabo la transmisión de datos, lo que se denomina canal *half-duplex*. El procedimiento *full-duplex* aún no se encuentra disponible. Sin embargo, la mayoría de los microprocesadores incorporan un canal de entrada y otro de salida y, como dos de los ocho contactos disponibles en la tarjeta están sin uso, se podría añadir una segunda vía de comunicación.

El intercambio de información, representado en la Figura 11, se realiza mediante unidades de protocolo APDU (*Application Protocol Data Unit*). Se emplean dos tipos de APDU: **comando** (Tabla 3), en la que se encapsula la información a transferir desde la aplicación externa a la tarjeta, y **respuesta** (Tabla 4), que incluye los datos enviados desde la tarjeta a la aplicación. Su formato está detallado y especificado en el ISO 7816-4 [15].

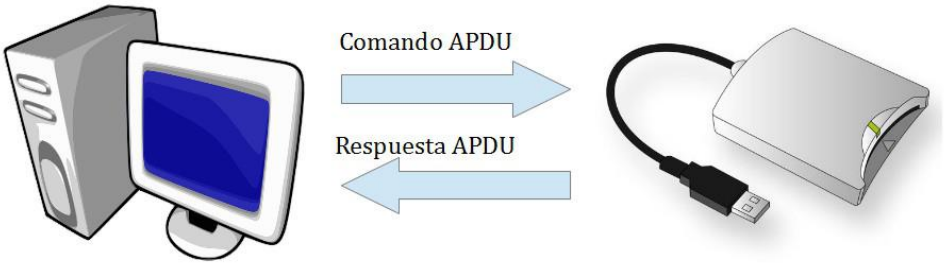

Figura 11. Ilustración de una comunicación PC-tarjeta.

Tabla 3. Estructura de un comando APDU.

CLA	INS	P1	P2	L_c	Datos	L_e

Campo	Longitud (byte)	Descripción
CLA	1	Indica la clase del comando
INS	1	Determina el comando enviado
P1	1	Parámetros específicos del comando enviado
P2	1	
L_c	0, 1 ó 3	Indica el número N_c de bytes del campo de datos
Datos	N_c	Longitud N_c de bytes de datos opcionales
L_e	0, 1, 2 ó 3	Indica el máximo número N_e de bytes de la respuesta

Tabla 4. Estructura de una respuesta APDU.

Datos	SW1	SW2

Campo	Longitud (byte)	Descripción
Datos	Como máximo N_e	Datos opcionales de respuesta
SW1-SW2	2	*Status Word*. Estado tras recibir y procesar el comando

Cada vez que se inserta una tarjeta en el lector, sus contactos se conectan a los del terminal y se procede a activarlos. La tarjeta inicia entonces su encendido y envía una respuesta llamada ATR (*Answer To Reset*) hacia el terminal. El ATR contiene información referente a cómo ha de ser la comunicación tarjeta-lector, estructura de los datos intercambiados, protocolo de transmisión, etc. Una vez que el lector interpreta el ATR, procede a enviar el primer comando. La tarjeta lo procesa y envía la respuesta asociada. Este intercambio de comandos y respuestas concluye una vez que la tarjeta es desactivada.

Las APDUs son transmitidas por el protocolo de nivel inferior a través de TPDUs (*Transmission Protocol Data Unit*). En el estándar ISO/IEC 7816-3 [16] se especifican varios protocolos de transmisión para este nivel, dos de los cuales son los más utilizados en la actualidad. El primero, T=0, es un protocolo orientado a byte (cada byte es transmitido de forma independiente). Se emplea un bit de paridad como método de detección de errores, siendo todo byte erróneo retransmitido. El segundo, T=1, es un protocolo orientado al bloque (paquetes de bytes). Este modo de intercambio permite control de flujo en ambas direcciones, con lo que la tarjeta puede tomar el control de la comunicación. T=0 es un protocolo cuyas ventajas residen

en su sencillez y es empleado por la mayoría de las aplicaciones. Los lectores implementan ambos protocolos, siendo usado el que se determine a partir del ATR.

2.4.4 Protocolo PC/SC

PC/SC (*Personal Computer/Smart Card*) [17] es un estándar para el uso de tarjetas inteligentes en sistemas operativos Windows, desarrollado por el *PC/SC Workgroup*, que trata de proporcionar una API de alto nivel para acceder a las tarjetas inteligentes y a los lectores de manera homogénea. Tiene la ventaja de que las aplicaciones no necesitan conocer los detalles de un lector para poder utilizarlo y de que una aplicación que sigue este estándar funciona con cualquier lector compatible con el mismo.

La estructura de la arquitectura de PC/SC representa la jerarquía de los programas y elementos de hardware, desde el usuario hasta la propia tarjeta. La aplicación es la que ofrece al usuario el interfaz de comunicación y accede al lector por medio del gestor de recursos, que se encarga de controlar qué lectores tiene disponibles el PC y si éstos tienen o no una tarjeta conectada. Para poder hacer uso de los lectores es necesario que el controlador correspondiente esté instalado, así como el del dispositivo de entrada/salida, pues un lector puede conectarse de diversas formas, ya sea de manera inalámbrica, USB o puerto serie.

2.4.5 Java Card

Java Card [18] es una tecnología que permite ejecutar de forma segura pequeñas aplicaciones basadas en Java, llamadas *applets*, en tarjetas inteligentes y otros sistemas embebidos. A pesar del alto grado de estandarización de las tarjetas inteligentes, tradicionalmente su funcionamiento interno depende del propio fabricante. La amplia variedad de posibilidades de las tarjetas inteligentes hace necesario superar esta falta de interoperabilidad y proporcionar una solución que impulse el desarrollo de aplicaciones que puedan ejecutarse sobre tarjetas de cualquier fabricante. La tecnología Java Card surge con este propósito, definiéndose como una plataforma segura y portable que incorpora las principales ventajas de Java.

Java Card define un entorno multiaplicación, en el que pueden coexistir diversas aplicaciones dentro del mismo chip de la tarjeta. En 1998, Visa introdujo la especificación *Visa OpenPlatform*, muy orientada a Java Card, que abrió a la comunidad en 1999 como *OpenPlatform*, actualmente gestionado y desarrollado por *GlobalPlatform* [19]. Esta especificación define la arquitectura de gestión de aplicaciones en una tarjeta inteligente multiaplicación, especificando la forma segura de carga e instalación de nuevas aplicaciones.

La limitada capacidad de memoria y de procesamiento de las tarjetas inteligentes, obligan a que Java Card únicamente soporte un restringido subconjunto de elementos del lenguaje Java, como se muestra en la Tabla 5.

Tabla 5. Elementos Java soportados y no soportados en Java Card.

Elementos Java soportados	Elementos Java no soportados
• Tipos de datos primitivos pequeños: `boolean, byte, short` • Arrays unidimensionales • Paquetes Java, clases, interfaces y excepciones • Orientación a objetos • El tipo de dato `int`, está opcionalmente soportado	• Tipos de datos primitivos grandes: `long, double, float` • Caracteres y *strings* • Arrays multidimensionales • Carga dinámica de clases • Gestor de seguridad • Recolector de basura • Multiproceso • Serialización de objetos • Clonación de objetos

Tratando de hacer la programación en Java de las tarjetas inteligentes lo más cómoda posible, Java Card proporciona cuatro paquetes con un API estandarizado:

- Paquete *java.lang* es la base del entorno Java para tarjetas inteligentes. Define las clases elementales y excepciones.
- Paquete *javacard.framework* sirve de complemento al anterior y proporciona funciones esenciales para la gestión de *applets*, así como su comunicación con el terminal, de acuerdo con la especificación ISO 7816-4 [15].
- Paquete *javacard.security* y su extensión *javacardx.cryto* aportan funciones para el trabajo con gran variedad de algoritmos criptográficos.

Además de éstos, también existen otros paquetes específicos para implementar ciertas aplicaciones, como, por ejemplo, para entornos de tarjetas SIM (*sim.access* y *sim.toolkit*).

2.4.6 Entorno de ejecución

La gestión, en tiempo de ejecución, de todos los recursos de la máquina virtual sobre la que se ejecuta Java se lleva a cabo a través del entorno de ejecución de Java (*Java Runtime Environment, JRE*). De la misma manera, la tecnología Java Card implementa un entorno de ejecución (*Java Card Runtime Environment, JCRE*). El JCRE se puede considerar el sistema operativo de una tarjeta Java Card, ya que se encarga de la gestión de los recursos de la tarjeta,

comunicaciones, ejecución de *applets*, seguridad, etc. Así, el JCRE incluye la máquina virtual, las clases del API (*Application Programming Interface*), las extensiones propias de cada fabricante y las propias clases inherentes de JCRE (Figura 12).

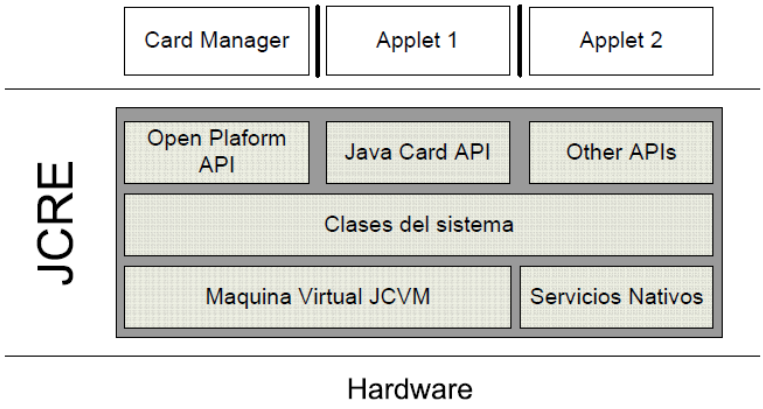

Figura 12. Arquitectura interna de un sistema Java Card.

El JCRE descansa sobre el hardware de la tarjeta, ocultando a las aplicaciones la tecnología propietaria subyacente y proporcionando un sistema estándar de desarrollo.

2.5 Tarjeta inteligente sin contacto

Del mismo modo que en muchas otras tecnologías, las tarjetas inteligentes también buscan la comodidad y la inmediatez que ofrece la tecnología inalámbrica. Además, cuentan con la ventaja de no necesitar baterías, ya que se activan y operan gracias a la energía que reciben del propio lector. A continuación, se exponen las principales características de este tipo de tarjetas.

2.5.1 Características físicas

Las tarjetas inteligentes que no necesitan el uso de sus contactos para la transmisión de información, son las tarjetas de proximidad o sin contactos (*contactless*). Utilizan una antena de cobre integrada en el cuerpo de la tarjeta, como se muestra en la Figura 13. Destacan los siguientes tres tipos de tarjetas en cuanto su conectividad:

- **EM4200**. Son tarjetas más antiguas y menos utilizadas actualmente. Tienen un chip de baja frecuencia de 125 KHz y se suelen utilizar en servicios que no requieren más de un dato.
- **ISO/IEC 15693**. Las tarjetas de vecindad (*vicinity cards*) fueron desarrolladas para solventar la ausencia de una tecnología de tarjetas inteligentes que pudiesen operar

a una distancia mayor de 10 centímetros, ya que ofrecen una distancia máxima de hasta 70 centímetros en su modo de lectura.

- **ISO 14443**. Es la tarjeta más extendida en la actualidad debido a su mayor versatilidad respecto a las anteriores, ya que puede ser utilizada para varias aplicaciones al mismo tiempo. Este tipo de tarjetas también trabajan a una frecuencia de 13.56 MHz y ofrecen una distancia máxima de lectura de 10 cm.

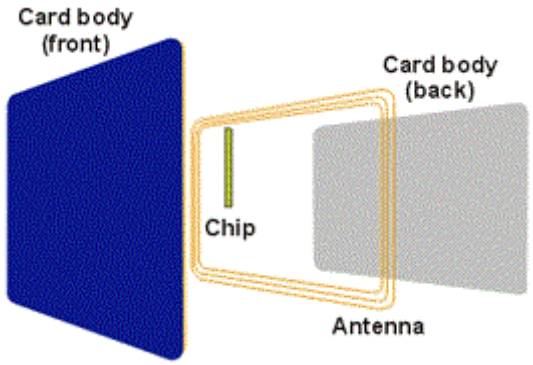

Figura 13. Antena integrada de una tarjeta.

Las tarjetas sin contactos se comunican con el lector mediante un proceso de transferencia de energía. Este proceso eléctrico es la transmisión de energía eléctrica inalámbrica entre dos bobinas acopladas magnéticamente para resonar a la misma frecuencia. Las tarjetas tienen sellados en su interior tres componentes: una bobina que hace la función de antena, un condensador y un circuito integrado. Cuando la tarjeta se coloca dentro del rango del lector, la antena y el condensador, absorben y almacenan la energía del campo. Esta energía se rectifica a corriente continua que es la que alimenta el circuito integrado de la tarjeta.

2.5.2 Tarjetas ISO 14443

La tecnología de tarjetas de proximidad descritas en el ISO/IEC 14443 [20] se utiliza para la gran mayoría de las implementaciones de tarjetas sin contacto en todo el mundo. Como ya se ha indicado, el rango de operación de este tipo de tarjetas está en torno a 10 cm, esta distancia varía en función de los requisitos de alimentación, tamaño de la memoria, procesador y coprocesador.

El estándar ISO/IEC 14443 se divide en cuatro partes:

- **Parte 1**: Características físicas. El tamaño de la tarjeta tiene el formato ya mencionado ID-1 (85.6 mm x 53.98 mm x 0.76 mm).
- **Parte 2**: Interfaz de potencia y señal de radio frecuencia. Esta sección describe las características técnicas del chip sin contacto, incluyendo parámetros de frecuencia, velocidad de datos, modulación y procedimientos de codificación de

bits. Dos variaciones se detallan en esta sección, la interfaz de tipo A y la de tipo B. Ambas operan a la misma velocidad y utilizan la misma velocidad de datos, pero se diferencian en la modulación y procesado de bits.

- **Parte 3**: Inicialización y anticolisión. La inicialización describe los requisitos para que el lector y la tarjeta establezcan una comunicación cuando ésta última entra en el campo de radiofrecuencia del lector. Anticolisión define lo que ocurre cuando varias tarjetas entran en un campo magnético al mismo tiempo, describiendo cómo el sistema determina con qué tarjeta comunicarse.
- **Parte 4**: Protocolo de transmisión. Esta sección define el formato de datos y los elementos de datos que permiten la comunicación durante una transmisión de información.

Para que un sistema cumpla con el estándar ISO/IEC 14443, es imprescindible que cumpla con los requisitos descritos en las cuatro partes.

Dentro del estándar se describen dos tipos de tarjetas, de tipo A y de tipo B. Aunque ambas trabajan a la frecuencia de 13.56 MHz, algunas de sus principales diferencias se encuentran en los métodos de modulación y codificación, en las velocidades de transmisión y en los protocolos de inicialización.

2.5.3 Protocolo PC/SC

A diferencia de las tarjetas de contacto, en las que la comunicación se inicia cuando la tarjeta envía un ATR al lector, para el caso de las tarjetas sin contacto que cumplen con el estándar ISO 14443-4, el lector solicita un ATS (*Answer To Select*) mediante el envío de un comando RATS (*Request for Answer To Select*). El comando RATS contiene dos parámetros imprescindibles para la posterior comunicación:

- El parámetro FSDI (*Frame Size Device Integer*) define el máximo número de bytes que deben ser enviados desde la tarjeta al lector en un solo bloque. Los posibles valores son: 16, 24, 32, …, 128 y 256 bytes.
- El parámetro CID (*Card Identifier*) es un identificador que se asigna a la tarjeta. De esta manera, resulta posible para un lector mantener una comunicación con varias tarjetas al mismo tiempo y dirigirse selectivamente a cada una de ellas a través de su CID.

Como respuesta al comando RATS, la tarjeta responde con un comando ATS y realiza la misma función que el ATR de las tarjetas de contactos. En esta respuesta, se definen los parámetros de protocolo del sistema operativo de la tarjeta, de manera que la transmisión de

información entre el lector y la tarjeta pueda ser optimizada en la relación a las propiedades de la aplicación implementada.

2.6 NFC (*Near Field Communication*)

NFC es una tecnología inalámbrica de corto alcance y alta frecuencia que permite el intercambio de información entre dos dispositivos o entre un dispositivo y una etiqueta/tarjeta NFC. Esta tecnología comenzó a desarrollarse en el año 2002 cuando Philips y Sony llegaron a un acuerdo con un protocolo compatible con las tecnologías sin contactos existentes en ese momento, *MIFARE* de *Royal Philips Electronics* y *FeliCa* de *Sony Corporation*[21]. Permite establecer comunicación con una separación máxima (real) aproximada de 5 cm. Para ello, NFC utiliza la inducción electromagnética entre dos antenas de espiras, operando en la banda ISM (*Industrial, Scientific and Medical*) de radiofrecuencia de 13.56 MHz descrito en el estándar ISO/IEC 18000-3 [22].

2.6.1 Estandarización

NFC es una tecnología descrita inicialmente por el protocolo NFCIP-1 (*Near Field Communication Interface and Protocol 1*). Está estandarizado según los ISO 18092 [23], ECMA 340 [24] y ETSI TS 102 190 [25], y especifica sus capacidades básicas, tales como velocidad de transferencia, esquemas de codificación de bits, modulación y protocolo de transporte. Además, se describen las precauciones necesarias para prevenir colisiones durante la fase de inicialización y los modos de operación activo y pasivo:

- **Activo**: Dispositivos que generan su propia señal de radio frecuencia (RF) para la transmisión de datos.
- **Pasivo**: Dispositivos que actúan únicamente como receptor sin generar ningún campo magnético propio.

Actualmente, los dispositivos NFC también implementan el protocolo NFCIP-2, definido en los estándares ISO 21481 [26], ECMA 352 [27] y ETSI TS 102 312 [28] y permite elegir entre los tres siguientes modos de operación:

- Transferencia de datos NFC (Protocolo NFCIP-1).
- Dispositivo de acoplo cercano (*PCD, Proximity Coupling Device*), definido en el ISO 14443.
- Dispositivo de acoplo en los alrededores (*VCD, Vicinity Coupling Device*), definido en el ISO 15693 [29].

Los dispositivos NFC han de facilitar estas tres funciones para asegurar la compatibilidad con los principales estándares internacionales de interoperabilidad entre tarjetas inteligentes, el ISO 14443 (tarjetas en condiciones de proximidad), el ISO 15693 (tarjetas en las inmediaciones) y el sistema de Sony para tarjetas inteligentes sin contacto, FeliCa. De este modo, NFC es compatible con la tecnología RFID, permitiendo su interacción con los millones de tarjetas inteligentes y otros dispositivos que existen por todo el mundo.

2.6.2 Interoperabilidad

Para alcanzar la meta de la interoperabilidad entre los diferentes dispositivos NFC, son necesarios distintos modos de operación, dependiendo de los dispositivos empleados.

- **Lectura/Escritura:** Se transmite información acorde a los estándares NDEF o RTD, por ejemplo, en aplicaciones de publicidad, en que se acerca un teléfono a un póster con un área resaltada que contiene un chip NFC.
- **Modo P2P:** Este modo de comunicación permite que dos dispositivos NFC se comuniquen entre sí para el intercambio de información.
- **Emulación de tarjeta:** Permite al dispositivo emular a una tarjeta NFC, permitiendo a otros dispositivos acceder y realizar operaciones.

2.6.3 Seguridad

Existen diferentes tipos de ataque y posibles medidas para mitigar sus efectos. A pesar de las limitaciones en rango de acción, como todo método de comunicación inalámbrico, NFC está sujeto a posibles escuchas de la señal RF o modificación de datos mediante el uso de generadores de interferencia.

Otro resquicio de seguridad existente son los ataques *relay*, en los que el usuario fraudulento sigue la petición de la unidad lectora hacia la potencial víctima y devuelve la respuesta de ésta a dicho lector en tiempo real, logrando usurpar su identidad.

Sin embargo, como NFC sienta las bases del protocolo de comunicación, serán los protocolos de capas superiores o las propias aplicaciones las que tengan la responsabilidad final de suministrar protección contra estas amenazas mediante el establecimiento de un canal seguro. Para lograr este propósito puede usarse sencillamente el acuerdo entre llaves *Diffie-Hellman*, mediante protocolos criptográficos como SSL, eliminando la amenaza de los ataques *Man-in-the-Middle*. Gracias a este canal seguro, NFC proporciona confidencialidad, integridad y autenticidad.

2.6.4 Aplicaciones

El número de aplicaciones de corto alcance atribuibles a la tecnología NFC crece continuamente, apareciendo ya en, prácticamente, todos los ámbitos de la vida diaria. La utilización de forma conjunta con teléfonos móviles ofrece grandes posibilidades:

- **Pago sin contacto**. El usuario realiza el pago sin que exista conexión entre el dispositivo que emplea y la máquina que cobra, como por ejemplo en autobuses.
- **Control de acceso**. El torno de acceso se activa aproximando un dispositivo NFC, como es el caso de ciertas empresas, que facilitan tarjetas con esta tecnología a sus empleados.
- **Conectividad**. NFC puede usarse para transmitir información de manera rápida acercando dos teléfonos, como Control de asistencia. Se puede emplear una etiqueta (*tag*) para registrar la asistencia a una clase o puesto de trabajo.
- **Aplicaciones médicas**. Existen aplicaciones de NFC en el ámbito sanitario, para seguimiento de pacientes, recetas, etc., como *Sesam-Vitale* [30] en Francia.

3 Implementación

Tras haber profundizado en los conceptos teóricos en el capítulo anterior, a continuación, se propone un sistema de autenticación de dos factores basado en la comunicación entre un servidor y un cliente utilizando un dispositivo móvil y una tarjeta. En este capítulo, se describe el diseño y la implementación de cada módulo que compone el sistema. A lo largo del mismo, se mostrarán ilustraciones que ayuden a entender mejor el funcionamiento del sistema implementado.

3.1 Arquitectura del sistema

En la Figura 14, se muestra una visión general de la estrategia de desarrollo tomada para este proyecto, que posteriormente se analizará paso a paso. Se trata de realizar un sistema de autenticación de dos factores de manera que como segundo factor se utilice una tarjeta inteligente conjuntamente con un terminal móvil. Para entender el concepto general del sistema, se observan dos elementos en el lado del cliente. Estos son el dispositivo móvil y la tarjeta. Por un lado, el dispositivo móvil es el encargado de comunicarse con el servidor y de ofrecer a la tarjeta la información necesaria. Por otro lado, la tarjeta es la encargada de generar un código TOTP necesario para la autenticación en el servidor.

El lado del servidor es el encargado de validar la autenticación del cliente en función de la información que recibe del mismo. A través del dispositivo móvil, el servidor recibe el código TOTP generado por la tarjeta. Para permitir la autenticación, el servidor genera su propio código TOTP y lo compara con el recibido por el cliente.

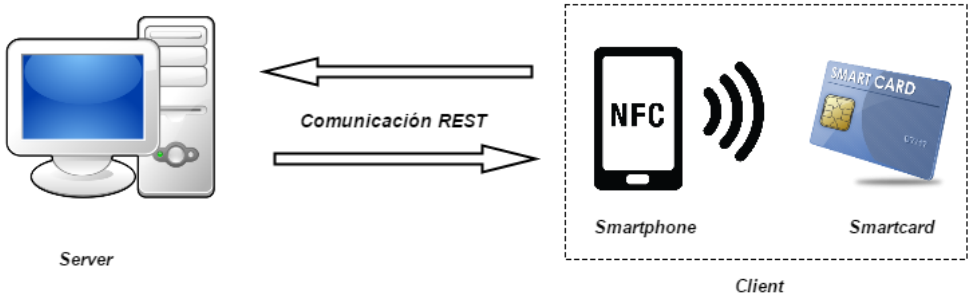

Figura 14. Arquitectura del sistema implementado.

El motivo por el cual se ha elegido el algoritmo TOTP frente al HOTP es, principalmente, por la mejora en la seguridad que supone. Al contrario que las contraseñas HOTP, en TOTP los códigos tienen una validez limitada a un corto periodo de tiempo. La gran desventaja de los sistemas TOTP es la necesidad de mantener sincronizados en el tiempo los dispositivos involucrados, en este caso el servidor y el terminal móvil que será quien se encargue de obtener el tiempo y suministrárselo a la tarjeta inteligente.

3.2 Implementación del sistema OTP

3.2.1 Generación del sistema local TOTP

Para comenzar con la implementación, se ha desarrollado un sistema de generación de códigos de un solo uso basada en el algoritmo TOTP descrito en el RFC 6238 [12]. Esta aplicación se ha implementado en Java utilizando el entorno de desarrollo Eclipse [31].

Como ya vimos en la sección 2.3.2, el algoritmo TOTP utiliza una clave secreta compartida y un contador (*factor móvil*). Como en este caso el factor móvil es el tiempo, el contador es representado como el número de intervalos de 30 segundos desde un instante inicial. Se ha elegido un intervalo de 30 segundos ya que resulta un valor equilibrado entre seguridad y usabilidad. Siguiendo el algoritmo descrito en el RFC 6238, para el desarrollo del cálculo del código del TOTP mediante el algoritmo HMAC, se ha utilizado la función de resumen criptográfica *SHA1*.

En la Figura 15, se muestra la estructura de los principales módulos de la implementación del sistema donde se destacan los métodos de generación del código TOTP, el cálculo del valor *HMAC-SHA-1* y la obtención del tiempo actual.

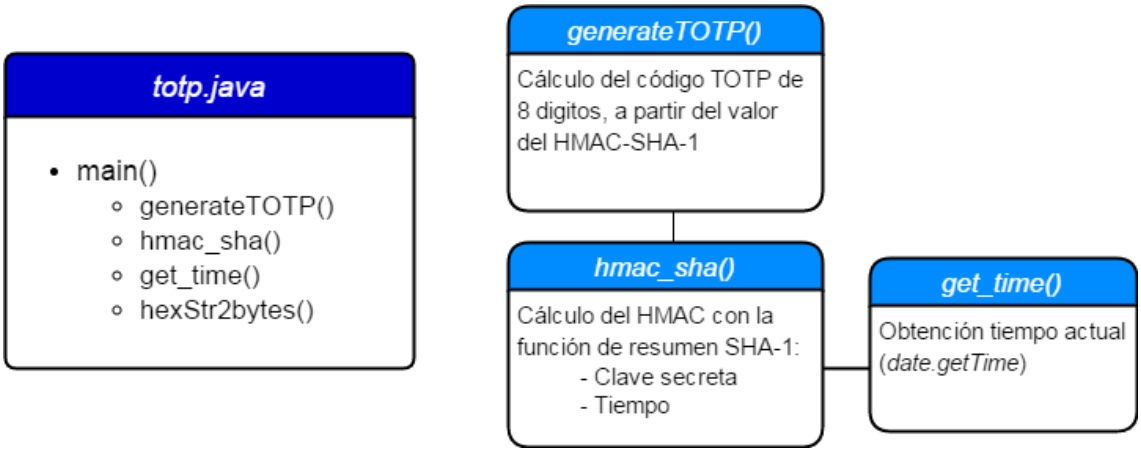

Figura 15. Estructura del sistema local TOTP.

En la Tabla 6, se muestran las salidas del primer programa implementado para tres valores de tiempo diferentes. En los resultados, se puede apreciar que el algoritmo se ha programado para tener una latencia con intervalos de 30 segundos, puesto que en los dos primeros resultados se genera un mismo valor de TOTP, debido a que se han realizado dentro de un mismo intervalo de tiempo de 30 segundos. Sin embargo, en el tercer resultado, el valor del TOTP ha cambiado como consecuencia de haberse realizado en el siguiente intervalo.

Tabla 6. Resultados del sistema implementado para tres valores de tiempo distintos.

Tiempo (seg)	Tiempo (UTC)	Valor de T (HEX)	TOTP	Modo
1480000000	2016-11-24, 15:06:40	00000002F0C455	**48163085**	SHA1
1480000005	2016-11-24, 15:06:45	00000002F0C455	**48163085**	SHA1
1480000031	2016-11-24, 15:07:11	00000002F0C456	**86671549**	SHA1

Estos valores contrastan la implementación en cuanto al tiempo, sin embargo, también se ha realizado la validación de los mismos a partir de los ejemplos que aparecen en el RFC 6238. Los resultados mostrados en la Tabla 7 han sido obtenidos a partir del programa implementado y coinciden con los valores del RFC mostrados en la Tabla 8.

Tabla 7. Resultados del sistema para diferentes tiempos.

Tiempo (seg)	Tiempo (UTC)	Valor de T (HEX)	TOTP	Modo
59	1970-01-01, 00:00:59	00000000000001	**94287082**	SHA1
1111111109	2005-03-18, 01:58:29	000000023523EC	**07081804**	SHA1
1234567890	2009-02-13, 23:31:30	0000000273EF07	**89005924**	SHA1

Tabla 8. Resultados de TOTP para SHA1 mostrados en el RFC 6238.

Tiempo (seg)	Tiempo (UTC)	Valor de T (HEX)	TOTP
59	1970-01-01, 00:00:59	0000000000000001	**94287082**
1111111109	2005-03-18, 01:58:29	00000000023523EC	**07081804**
1111111111	2005-03-18, 01:58:31	00000000023523ED	**14050471**
1234567890	2009-02-13, 23:31:30	000000000273EF07	**89005924**
2000000000	2033-10-11, 03:33:20	0000000003F940AA	**69279037**
20000000000	2603-10-11, 11:33:20	0000000027BC86AA	**65353130**

3.2.2 Migración a entorno servidor

Tras haber realizado una primera aplicación que genera códigos TOTP, el siguiente paso ha sido migrar la solución a un entorno conectado que nos permita validar los códigos en cualquier momento y, por tanto, autenticar satisfactoriamente a un determinado usuario o entidad.

Para ello, se ha desplegado un servidor Apache Tomcat sobre el que se ejecuta el servicio web de generación y validación del código TOTP. Para la comunicación entre un cliente y el

servidor, se ha elegido una arquitectura REST (*Representational State Transfer*) utilizando directamente el protocolo HTTP (*HyperText Transfer Protocol*) para obtener los datos o indicar la ejecución de operaciones.

REST es un estilo de arquitectura de desarrollo web que permite crear servicios y aplicaciones que pueden ser usadas por cualquier dispositivo o cliente que implemente el protocolo HTTP, por lo que es notablemente más simple y convencional que otras alternativas que se han empleado en los últimos años, como SOAP (*Simple Object Access Protocol*) o WSDL (*Web Services Description Language*).

La comunicación cliente-servidor es sin estado, esto es, trata cada petición como una transacción independiente sin importar cualquier solicitud anterior, de modo que la comunicación se compone de pares independientes de solicitud y respuesta (Figura 16). Cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes.

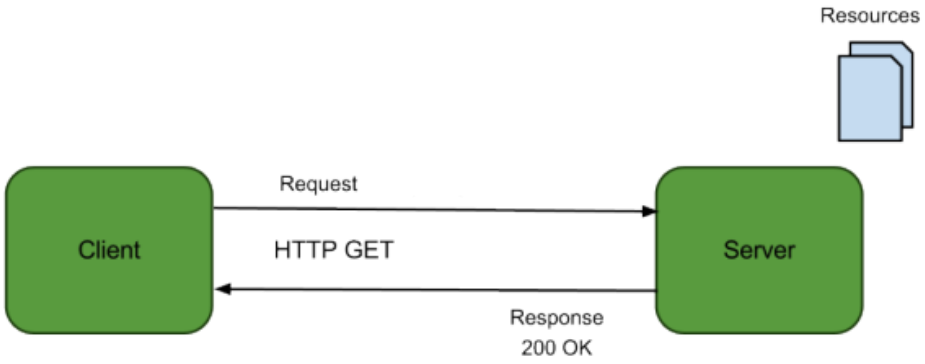

Figura 16. Petición GET cliente-servidor

En la comunicación REST, dado un URI *(Uniform Resource Identifier)*, y, mediante el protocolo HTTP, podemos operar sobre los recursos del servidor. La operación a realizar se especifica mediante los métodos estándar de HTTP:

- El método **GET** en una petición, se utiliza para consultar y leer recursos.
- El método **POST** requiere al servidor aceptar la petición para crear recursos.
- El método **PUT** sirve para editar un recurso ya existente en el servidor. Si el recurso no existe, el servidor puede crear uno nuevo.
- El método **DELETE** se utiliza para eliminar el recurso especificado.

La lógica del servidor se divide en dos grandes partes: por un lado, la implementación del algoritmo para el cálculo de la contraseña basada en tiempo (TOTP) mediante el algoritmo HMAC [10] utilizando la función hash criptográfica *SHA1*, y, por otro lado, la implementación de la interfaz de comunicación a través de un API REST.

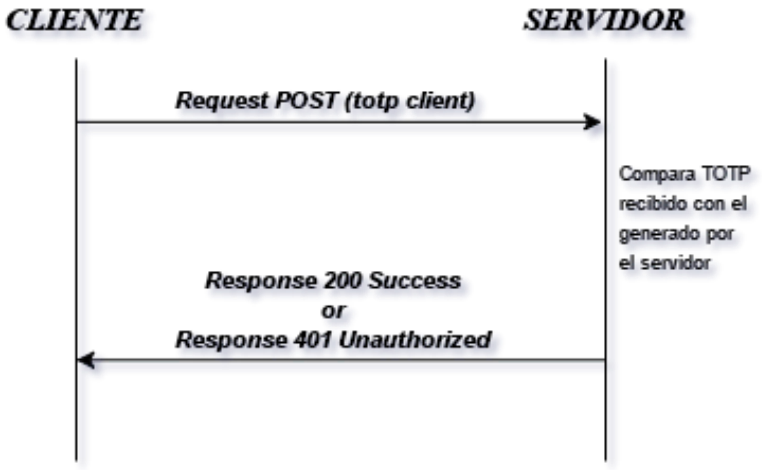

Figura 17. Intercambio de comandos entre el cliente y el servidor.

El intercambio de información se muestra en el esquema de la Figura 17. El proceso comienza cuando el terminal móvil inicia la comunicación enviando hacia el servidor, mediante el identificador de recursos (URI), una petición con el método POST en el que se incluye el código TOTP. El servidor, al recibir estos datos, se encarga de generar su código TOTP y compararlo con la información recibida. Seguidamente, en función de la validez del código recibido, el servidor envía la respuesta hacia el terminal móvil con el resultado satisfactorio o desfavorable.

Para la comprobación del correcto funcionamiento de la comunicación entre el cliente y el servidor, se ha capturado el tráfico de tramas mediante el analizador de protocolos *Wireshark* [32]. En las Figuras 18 y 19, se muestran dos ejemplos de peticiones HTTP POST: el primero, en el que el código TOTP enviado es correcto, la respuesta es satisfactoria retornando un *200 OK*; y seguidamente, el segundo, donde se envía intencionadamente un código TOTP incorrecto, la respuesta es desfavorable retornando un *401 Unauthorized*.

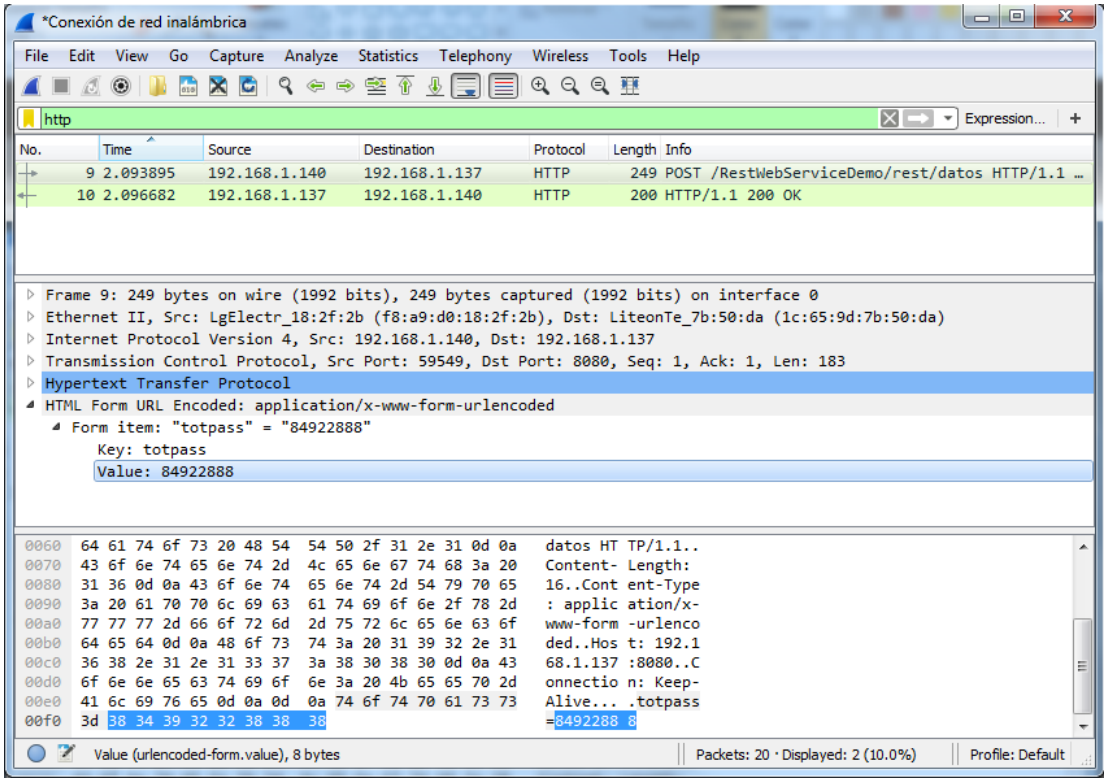

Figura 18. Captura de tráfico de una petición HTTP POST con un TOTP correcto.

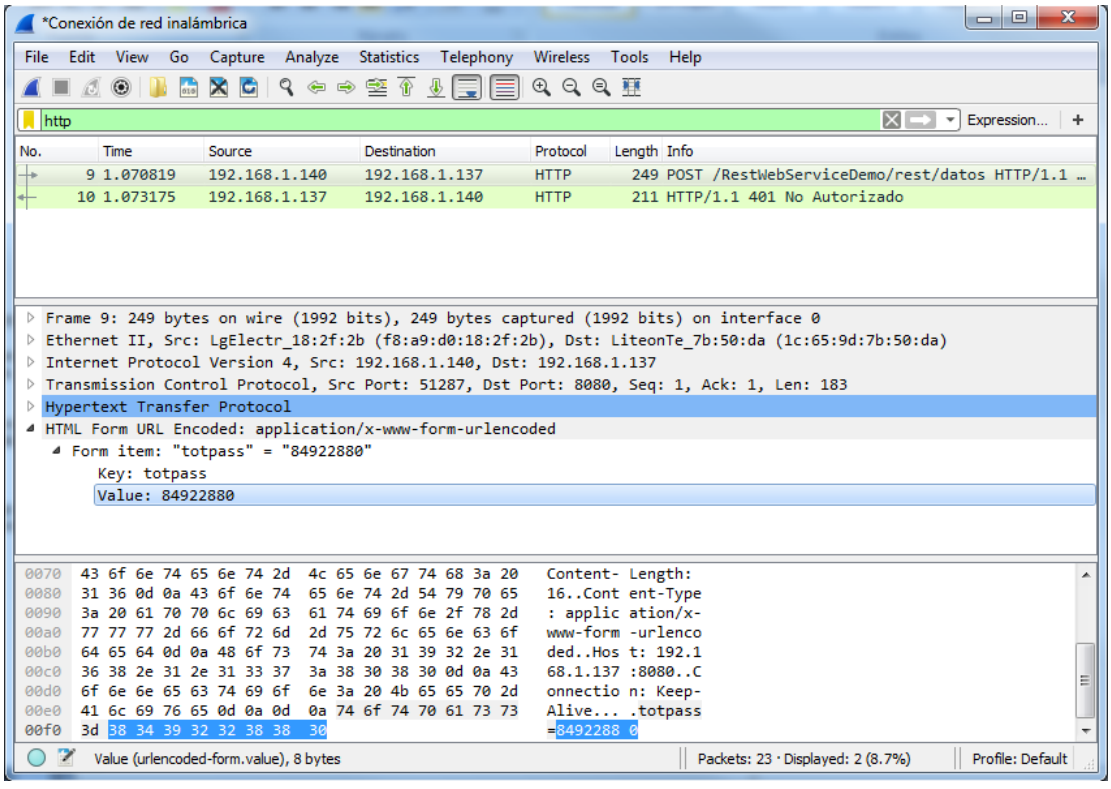

Figura 19. Captura de tráfico de una petición HTTP POST con un TOTP incorrecto.

3.2.3 Desarrollo de aplicación móvil como cliente del servidor

Una vez realizado el sistema cliente-servidor, se ha desarrollado una aplicación móvil que mimetice el comportamiento del cliente anteriormente desarrollado en entorno de PC. Esta aplicación móvil se ha implementado en Android, empleando el entorno de desarrollo Android Studio en su versión 2.2.

La aplicación puede dividirse en dos niveles. Por un lado, está el diseño de la interfaz de usuario compuesto por el *layout* y por *views*. El *layout*, definido en un documento XML, incluye la estructura y el orden de los elementos para que el usuario pueda interactuar con la interfaz. Dentro de éste, se encuentran los *views*, que son los elementos (*Labels*, *Buttons*, *TextFields*, etc.) que permiten controlar la interacción del usuario con la aplicación. En la Figura 20, podemos el *layout* de la aplicación en el proceso de implementación.

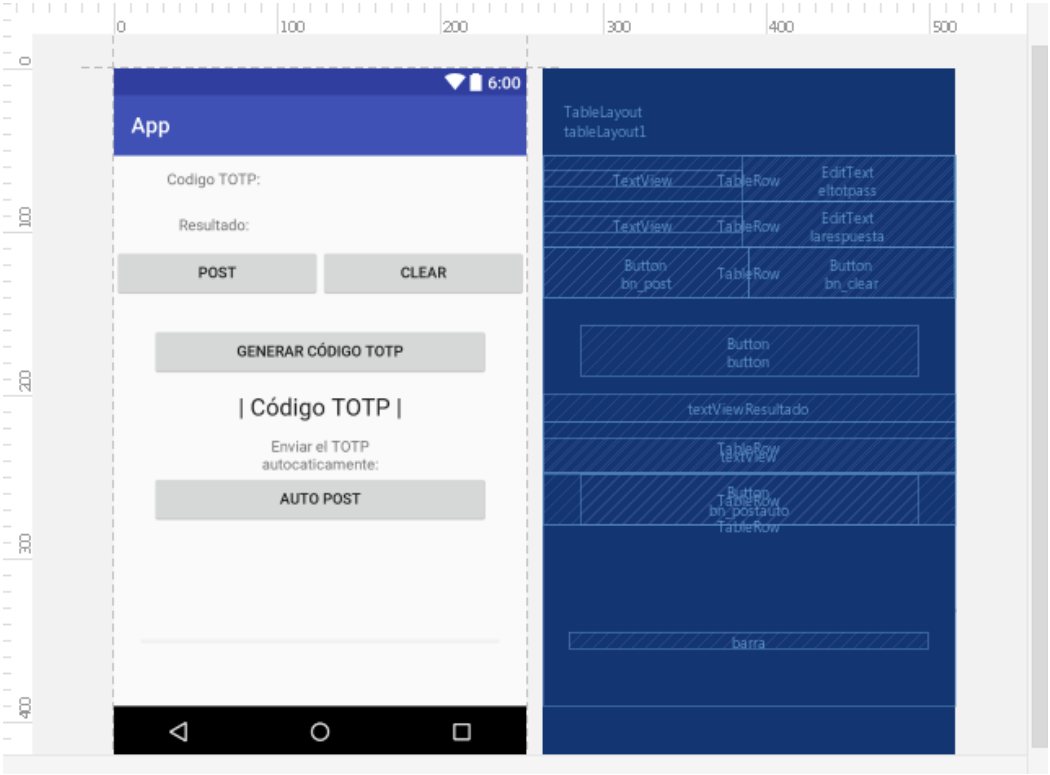

Figura 20. Layout de la aplicación móvil.

Por otro lado, se encuentra la parte correspondiente a la lógica de la aplicación. Los principales módulos correspondientes son los mostrados en la Figura 21 y cuyas funcionalidades son las siguientes:

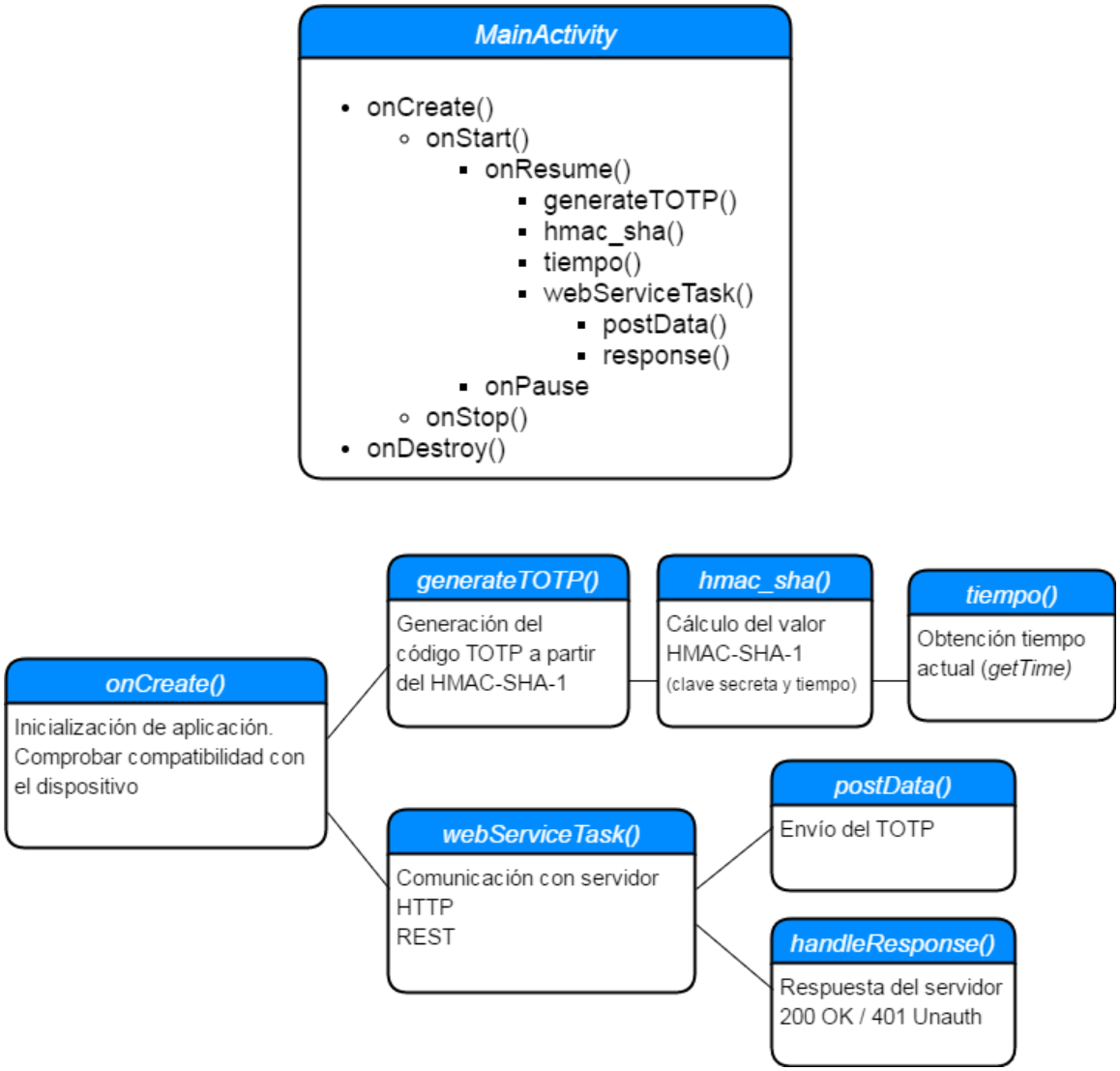

Figura 21. Estructura del programa principal.

- **onCreate:** En este método se inicia la actividad del programa. Se definen los recursos necesarios para la interfaz de usuario, mostrada en la Figura 22, y los *widgets*, empleando *findViewById(int)*, necesarios para interactuar con el usuario. Entre ellos, el botón para generar el TOTP y la barra de tiempo de 30 segundos en la parte inferior.
- **tiempo**: Realiza la obtención del tiempo Unix. Devuelve el tiempo transcurrido en milisegundos desde el 1 de enero de 1970 hasta el momento almacenado en el objeto *fecha* al que se aplica.
- **hmac_sha**: Método que realiza la función del mecanismo criptográfico *HMAC* con la función de resumen *SHA1* a partir de la clave secreta y el tiempo actual obtenido en el método tiempo().
- **generateTOTP**: A partir del valor del *HMAC-SHA-1*, este método realiza la función de truncado a un número entero de 8 dígitos, que será el código TOTP.
- **webServiceTask()**: En este método se realiza la comunicación REST con el servidor.

- **postData**: Envío del TOTP mediante una petición POST directamente a la dirección URL del servidor.
- **handleResponse**: Se encarga de recibir la respuesta del servidor, pudiendo ser satisfactoria (*200 OK*) o desfavorable (*401 Unauthorized*).

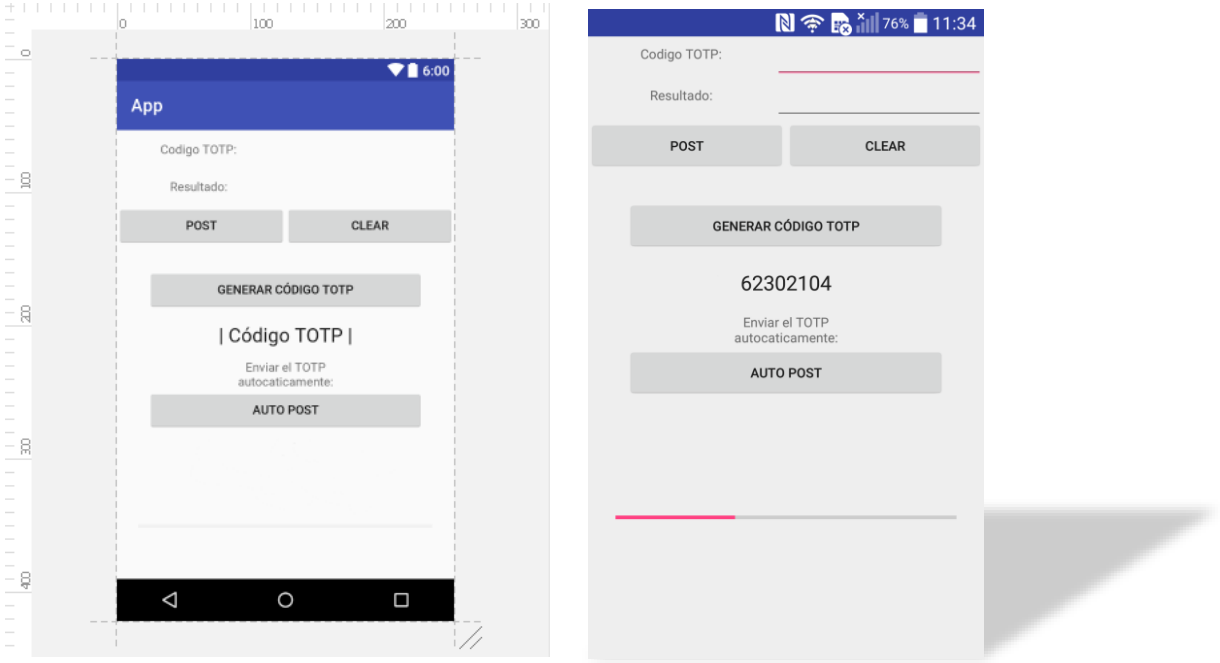

Figura 22. Interfaz de usuario de la aplicación.

En la Figura 22, se muestra la interfaz de usuario de la aplicación. A la izquierda de la imagen, el diseño en Android Studio y a la derecha una captura de pantalla de la aplicación ejecutando en el dispositivo móvil, con un código TOTP generado. Para comprobar la validez del dicho código, se compara con el valor marcado de la Tabla 9, cuyo resultado pertenece a la salida del cliente implementado anteriormente de forma local.

Tabla 9. Resultado del cliente implementado localmente

Tiempo (seg)	Tiempo (UTC)	Valor de T (HEX)	TOTP	Modo
1481456091	2016-12-11, 11:34:51	00000002F181EED	**62302104**	SHA1

3.3 Soporte OTP sobre tarjeta inteligente.

La tarjeta inteligente, debido a su robustez y seguridad, resulta el dispositivo más apropiado para tratar de reforzar la seguridad del sistema. Para ello, en lugar de realizar la generación del OTP en el dispositivo móvil, se va a tratar de trasladar la ejecución del algoritmo criptográfico dentro de la tarjeta, de manera que la clave secreta utilizada para el cálculo del TOTP estará fuertemente custodiada en un dispositivo seguro.

3.3.1 Entorno de tarjeta inteligente

Desarrollar sistemas basados en tarjetas inteligentes multiaplicación es un reto debido a que se debe soportar la ejecución de un gran número de aplicaciones en cualquier momento, así como ofrecer la posibilidad de actualizar o eliminar dichas aplicaciones e instalar otras nuevas cuando se estime oportuno.

GlobalPlatform [19] ofrece un entorno donde realizar todas estas operaciones de gestión de forma segura y eficiente. El control de la tarjeta está en manos de su emisor, pero éste puede otorgar derechos de gestión a terceros. A través del *Card Manager*, se realizan las principales funciones de administración de la tarjeta, como la instalación, selección, transferencia de comandos a la aplicación adecuada, etc., además de la definición de los dominios de seguridad y el establecimiento de los canales seguros de comunicación entre la tarjeta y el lector. GlobalPlatform define un conjunto de comandos o API a través de los cuales se puede acceder a todas esas funcionalidades.

La tarjeta inteligente que se ha utilizado para este sistema implementa el estándar ISO/IEC 14443 Tipo A [20]. Se trata de una tarjeta dual del fabricante Gemalto modelo Optelio Contactless R7. Implementa soporte para la especificación Java Card 2.2.2 [33] y GlobalPlatform 2.1.1.

3.3.2 Desarrollo de applet Java Card

Para la gestión de las aplicaciones en la tarjeta, como se ha indicado, se sigue la especificación GlobalPlatform. Se ha empleado la aplicación de código libre GlobalPlatformPro [34] como cliente de comunicación con la tarjeta, pues soporta los algoritmos de seguridad y todo el conjunto de comandos de gestión necesarios para instalar, instanciar y borrar aplicaciones de una tarjeta Java Card. Adicionalmente, como apoyo en el desarrollo de la propia aplicación, se ha utilizado el *plug-in* de Java Card disponible para Eclipse. Ambos elementos han sido complementarios, puesto que por un lado el primero permite realizar la validación del desarrollo en una plataforma hardware completa y por el otro, el segundo permite un desarrollo rápido y ágil, aunque de manera emulada.

Dado que se ha dispuesto que la tarjeta almacene la clave secreta para la generación del OTP de forma segura, se entiende que la tarjeta debería implementar igualmente los procedimientos para generar el OTP y que así la clave no se vea nunca comprometida por ser utilizada fuera del dispositivo seguro.

Sin embargo, la tarjeta no tiene incluida la librería necesaria para el cálculo directo del *HMAC-SHA-1*. La implementación de la clase *javacard.security.Signature* [35] y sus métodos

correspondientes a la obtención directa del HMAC no están disponibles en la tarjeta que se ha empleado en este proyecto. Por ello, ha sido necesaria su implementación a través de la clase *javacard.security.MessageDigest* [36], uno de cuyos métodos permite la obtención de la función *SHA1*. Esta opción, aunque no óptima como sería el cálculo incluido en el propio sistema operativo de la tarjeta, dota a la aplicación de la funcionalidad deseada. De esta forma, se ha seguido el funcionamiento del algoritmo HMAC descrito en el Anexo I.

Para la posterior obtención del código TOTP, es necesario el valor del tiempo actual. La tarjeta no dispone de un reloj interno y por tanto no puede obtener este valor de manera autónoma. Por consiguiente, se debe proporcionar el valor del tiempo actual de una fuente externa cada vez que se desee calcular el código TOTP. La solución empleada consiste en realizar el envío del tiempo desde un dispositivo en el campo de datos de un comando APDU previamente definido.

Disponible el valor del tiempo, la tarjeta ya puede realizar el cálculo del TOTP a partir de la clave secreta almacenada. En la aplicación implementada, se permite almacenar hasta cinco claves distintas que se guardan en un *array* unidimensional de 100 bytes de longitud. Para la gestión y actualización de las claves de utilizan los parámetros P1 y P2 de los comandos APDU, cuya estructura se mostrará seguidamente en la sección 3.3.2.1.

A continuación, en la Tabla 10, se muestran los métodos empleados por el JCRE (sección 2.4.6) del *applet* implementado e instalado en la tarjeta:

Tabla 10. Métodos empleados en el applet.

Applet javicard.java
• *install()* • *process()* • *set_key()* • *generateHash1()*

- **Instalación y registro – *install***: El JCRE llama a este método para crear una instancia e inicialización del *applet* en la tarjeta.
- **Intercambio de APDU – *process***: Método en el que se recibe y procesa el APDU entrante. Desde aquí, si el campo INS del APDU es igual a 0x50, se llama al método *generateHash1()*. En cambio, si es 0x60, se llama al método *set_key*().
- **Almacenamiento de nuevas llaves - *set_key***: Método que realiza el guardado de una posible nueva llave secreta. Enviada en el campo de datos del comando APDU

y dependiendo de los parámetros P1 y P2, se podrán almacenar hasta cinco llaves diferentes de 20 bytes de longitud (*long_key*). El parámetro P2 indicará en qué posición se guarda cada una de las claves (*P2*long_key*).

- **Cálculo del TOTP - *generateHash1***: En este método, se realiza la generación del TOTP. Primero, se realiza el cálculo del *HMAC-SHA-1*, utilizando la clase *MessageDigest* para el algoritmo del *SHA1*, obteniendo un valor de 20 bytes. Seguidamente, con ese valor, se realiza el truncamiento para obtener el código TOTP de 8 dígitos.

3.3.2.1 Gestión de comandos APDU

Para la comunicación con la tarjeta se han definido dos tipos de comandos APDU: de tipo **GENERATE_TOTP** y de tipo **SET_KEY**.

El comando GENERATE_TOTP, cuya estructura se muestra en la Tabla 11, realiza el cálculo del código TOTP. Este valor se puede obtener utilizando una de las cinco posibles claves almacenadas en la tarjeta, indicando cuál utilizar mediante el valor del parámetro P2. En el campo de datos del APDU se envía el valor del tiempo, necesario para el HMAC-SHA-1.

Tabla 11. Estructura del comando de tipo GENERATE_TOTP.

Código	Valor	Descripción
CLA	00	Comando ISO/IEC 7816-4
INS	50	Cálculo de TOTP
P1	0	Parámetro P1
P2	{0…4}	Parámetro P2, indica qué clave emplear
Lc	‘XX’	Longitud del campo de datos
Datos	‘XXX…’	Tiempo actual
Le	7F	Longitud máxima de la respuesta esperada

El comando de tipo SET_KEY, cuya estructura se muestra en la Tabla 12, envía los datos necesarios para realizar una posible actualización o almacenamiento de nuevas claves secretas en la tarjeta. Como se ha mencionado anteriormente, se pueden almacenar hasta cinco claves distintas, por lo que se podrá seleccionar cuál es la clave que se quiere actualizar mediante el valor del parámetro P2. En este caso, en el campo de datos del APDU se envía el valor de esta nueva clave a almacenar.

Tabla 12. Estructura del comando de tipo SET_KEY.

Código	Valor	Descripción
CLA	00	Comando ISO/IEC 7816-4
INS	60	Nueva clave secreta
P1	0	Parámetro P1
P2	{0…4}	Parámetro P2, indica qué clave actualizar.
Lc	'XX'	Longitud del campo de datos
Datos	'XXX…'	Nueva clave secreta a almacenar
Le	7F	Longitud máxima de la respuesta esperada

Una ejecución satisfactoria de cualquier comando enviado se indicará empleando una respuesta APDU con un código de estado (*status word)* de valor 0x9000. En caso de que el comando sea erróneo, el *status word* tomará uno de los valores listados en el ISO/IEC 7816-4 [15], donde algunos de los más habituales se muestran en la Tabla 13.

Tabla 13. Mensajes de errores de respuesta más habituales.

SW1	SW2	Descripción
67	00	Longitud Lc incorrecta
69	85	Condiciones de uso de APDU incorrectas
69	99	Comando desconocido
6A	86	Parámetros P1 y P2 incorrectos
6D	00	Código de instrucción incorrecto
6F	00	Error no diagnosticado

Para la comprobación y testeo del correcto funcionamiento del *applet*, se ha empleado la utilidad *PC/SC Scriptor* [37]. Es una aplicación creada por la empresa francesa SpringCard y está diseñada para establecer una comunicación entre un ordenador y una tarjeta inteligente utilizando un lector de tarjetas USB. Está diseñada para funcionar tanto con tarjetas de contacto como tarjetas *contactless* mediante comunicación NFC. Esta aplicación permite enviar y recibir comandos APDU a una tarjeta inteligente, pudiéndose realizar de manera manual o automáticamente mediante la lectura de un archivo *script.*

A continuación, en la Figura 23, se muestra una captura de un ejemplo de comunicación empleando esta utilidad, en la ventana de entrada, *Script*, se introduce el valor en hexadecimal de los APDU a enviar y en la ventana de salida, *Output*, se muestra el valor de los APDU de respuesta.

En el ejemplo, se observa el envío de dos comandos APDU: el primero, de tipo selección del applet y el segundo, de tipo GENERATE_TOTP. Como respuesta al primero, se recibe un APDU con el *status word* 0x9000 indicando que se ha seleccionado correctamente el *applet*. Y como respuesta al segundo, se recibe un APDU satisfactorio en el que dentro del campo de datos se incluye el valor del TOTP calculado por la tarjeta.

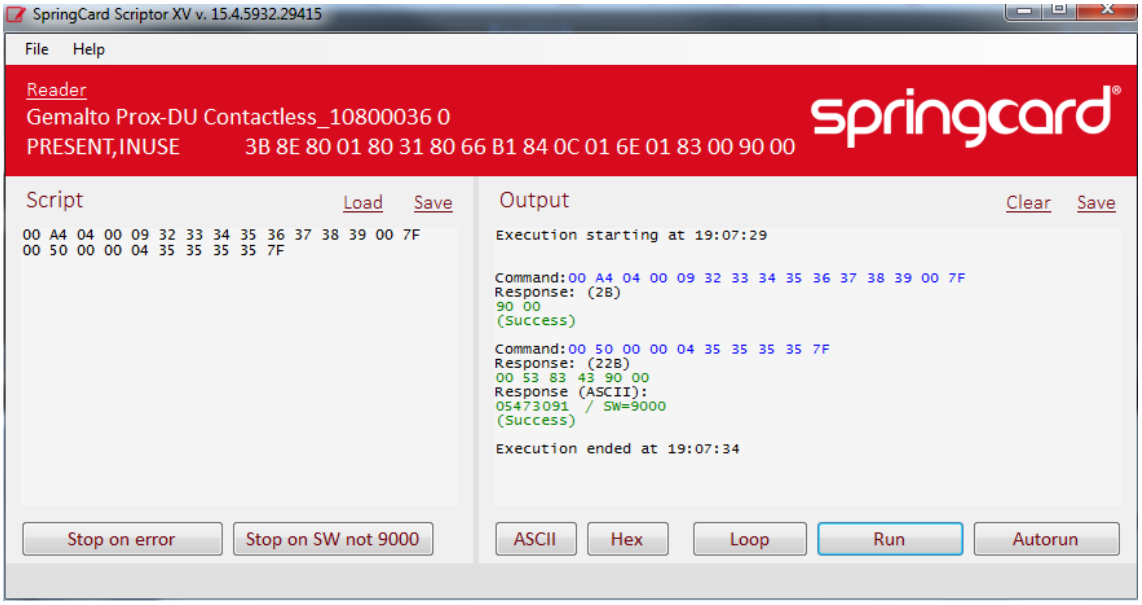

Figura 23. Envío y respuesta de un comando GENERATE_TOTP.

El correcto funcionamiento del cálculo del TOTP se ha comprobado comparando el resultado calculado por la tarjeta con el calculado por el sistema local implementado anteriormente. El valor recibido en el campo de datos de la respuesta APDU mostrado en la Figura 23 (**05473091**) coincide con el calculado de forma local utilizando los mismos parámetros de clave secreta y tiempo de la Tabla 14.

Tabla 14. Resultado del sistema OTP local para un tiempo determinado.

Tiempo (seg)	Tiempo (UTC)	Valor de T (HEX)	TOTP	Modo
5555	1970-01-01, 01:32:35	000000000000B9	**05473091**	SHA1

3.3.3 Integración final con aplicación móvil

Inicialmente, la aplicación móvil se había desarrollado para que el cálculo del código TOTP lo generase el propio terminal, sin embargo, para las tareas de asegurar el almacenamiento de la clave secreta y generación del TOTP, se ha elegido un dispositivo seguro como es la tarjeta inteligente.

Para completar la implementación del sistema, se ha integrado el *applet* desarrollado en la sección anterior con la aplicación móvil. Se ha adaptado la aplicación móvil para ser compatible con una transmisión NFC y lograr comunicarse con la tarjeta.

En la Figura 24, se observa la nueva estructura de la lógica de la aplicación móvil donde se ha incluido el nuevo método *onNewIntent()*, cuya funcionalidad es la de realizar dicha comunicación NFC. En este método, empleando la clase IsoDep [38] y mediante la operación *transceive*, se procede a enviar los APDUs correspondientes de selección (SELECT [15]) y de generación (GENERATE_TOTP), previamente definidos. En este último, en su campo de datos, se incluye el valor del tiempo actual.

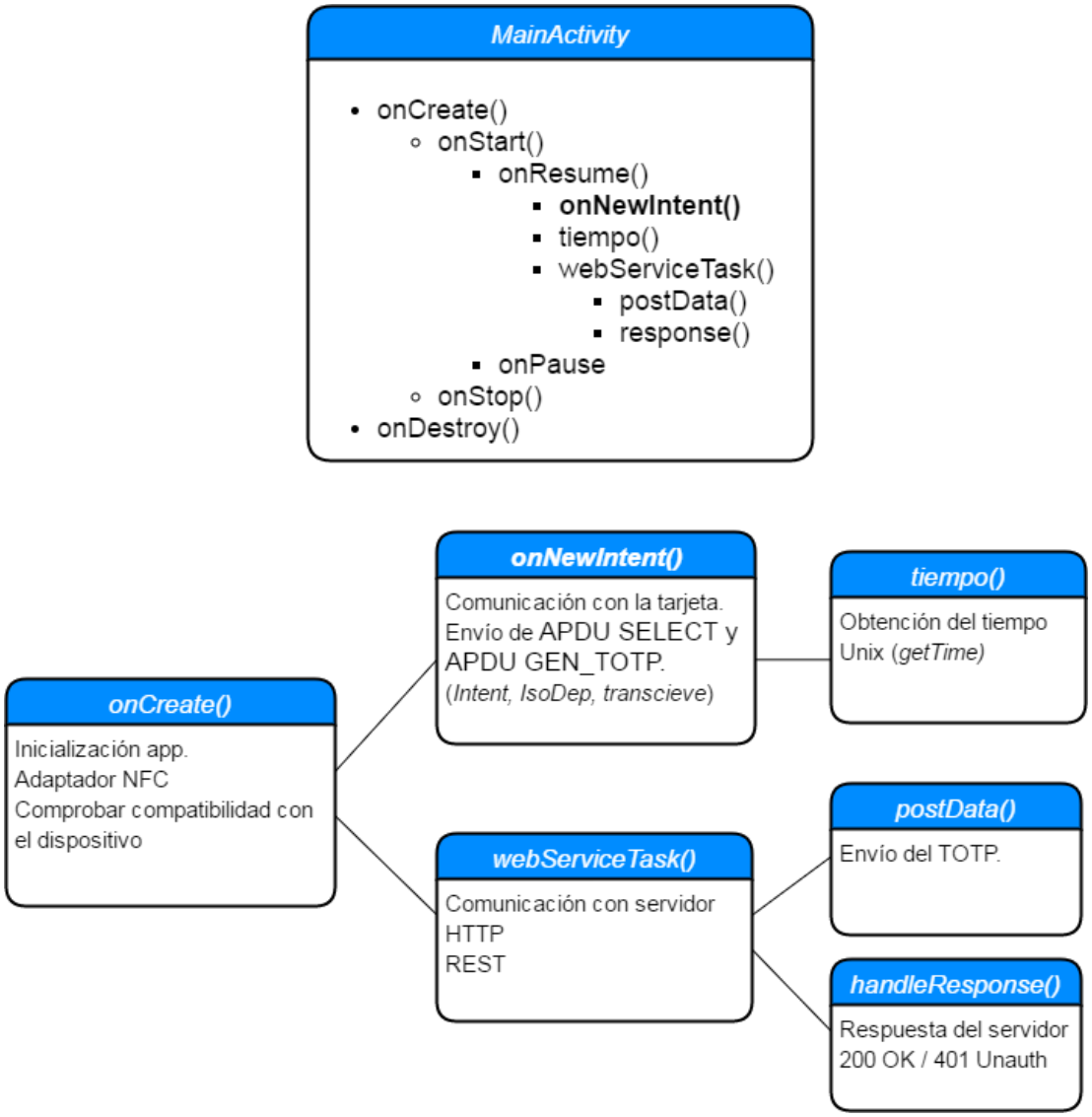

Figura 24. Estructura del nuevo programa principal.

Al igual que en la lógica de la aplicación móvil, también ha sido necesario modificar la interfaz de usuario (*layout*) de la misma, mostrada en la Figura 25. Se ha incluido un mensaje

donde se informa al usuario la necesidad de acercar la tarjeta al dispositivo móvil, así como un campo de datos para mostrar el resultado de la operación.

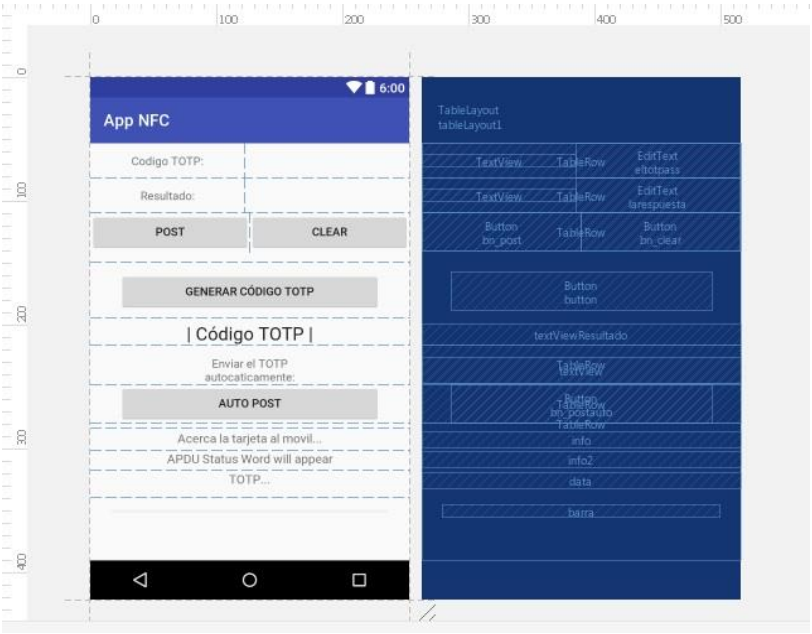

Figura 25. Layout de la aplicación con soporte NFC.

En la Figura 26, se muestra el proceso de actividad del *applet* en la tarjeta y el intercambio de APDUs entre el móvil y la misma. Esta comunicación consiste en el envío de los APDUs de selección (SELECT) y de generación (GENERATE_TOTP), con sus correspondientes respuestas del servidor incluyendo, en la segunda, el valor del TOTP calculado.

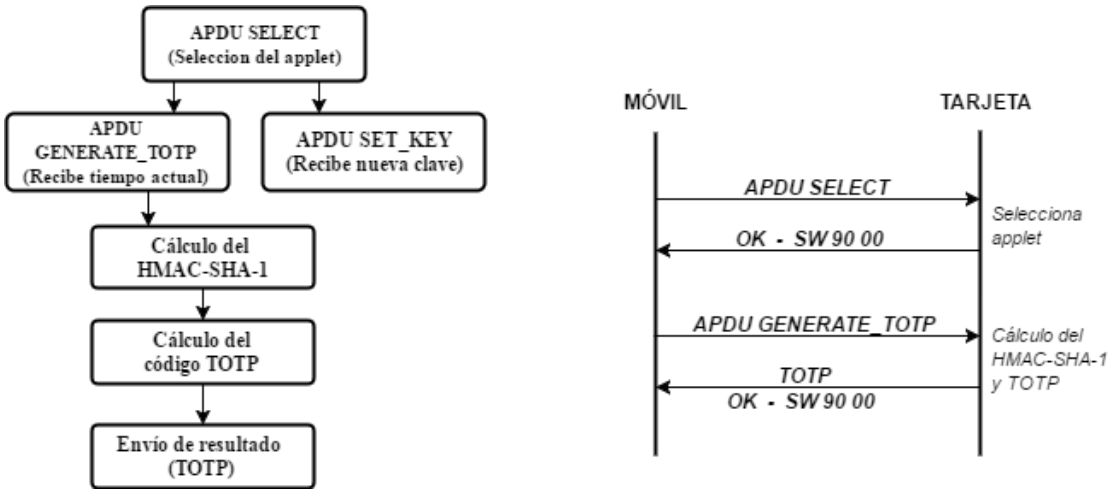

Figura 26. Estructura del applet e intercambio de APDUs entre ambos dispositivos.

Las primeras pruebas reportaban un error debido a la duración del cálculo del *HMAC-SHA-1* por parte de la tarjeta. Este exceso de tiempo provocaba que se produjera una desconexión entre la tarjeta y el móvil, lo que causaba el error. Para solventarlo, se ha elevado el

tiempo de espera de la operación *transceive* desde la aplicación de Android mediante un *setTimeout()* de 2800 ms.

Para comprobar el correcto funcionamiento y la validez del código TOTP, se ha comparado el valor que se generaba íntegramente en el dispositivo móvil con el que ahora se genera al acercar la tarjeta. En la Figura 27, se muestra, en una imagen, el resultado de ambos valores. En este ejemplo, estos valores se muestran por pantalla con el único objetivo de comprobar los resultados, ya que, en la aplicación real final, por seguridad, el usuario no tendría acceso directamente al código TOTP.

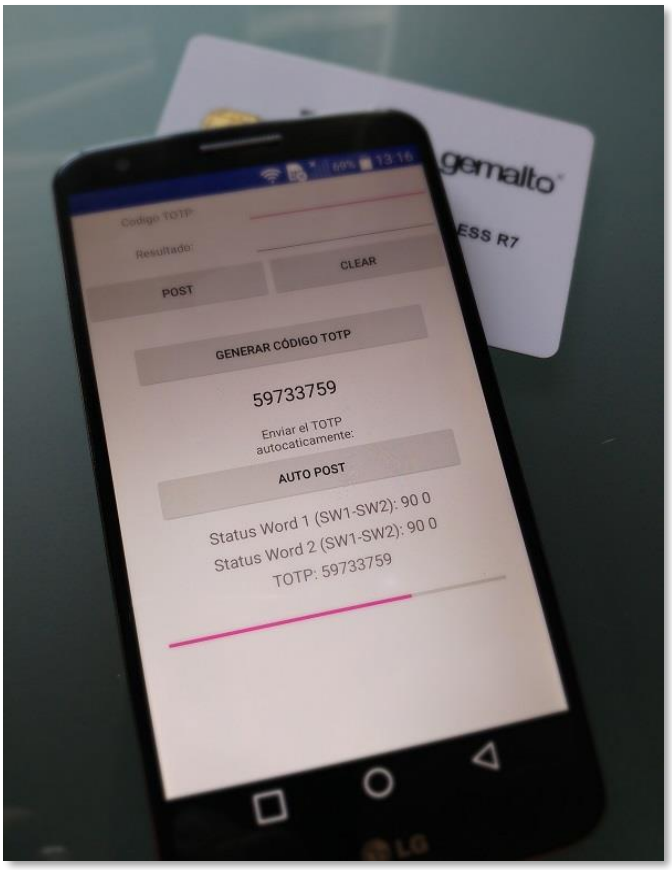

Figura 27. Valor del TOTP generado al acercar la tarjeta al dispositivo móvil.

Finalmente, los distintos elementos de la arquitectura descritos en los apartados anteriores han sido integrados en un solo sistema cuya estructura fue mostrada en la Figura 14. Las características particulares de cada uno de los recursos empleados en la implementación, tanto software como hardware, son las señaladas en el documento Anexo II.

No obstante, el prototipo implementado es poco sensible a las distintas versiones de las herramientas empleadas para su implementación. Existen algunas excepciones, como es el caso de la tarjeta y su versión Java Card donde la implementación del algoritmo *SHA1* es

dependiente de la librería disponible en la propia tarjeta, es decir, la no existencia del método *Digest* obligaría a la modificación del código Java Card.

Las pruebas de funcionamiento del sistema completo han ido dirigidas a comprobar su correcto funcionamiento modificando los diferentes parámetros que podrían variar en un sistema real, tales como claves, tiempos, etc. Únicamente, se ha buscado que la aplicación completa se ejecute correctamente, sin ninguna restricción sobre el rendimiento. De hecho, bajo las condiciones actuales, solamente el tiempo empleado por la tarjeta para llevar a cabo el algoritmo *SHA1* podría considerarse elevado.

3.4 Algunos aspectos sobre las vulnerabilidades

Aunque en este trabajo no se ha dedicado especial atención a las vulnerabilidades del mecanismo TOTP implementado, conviene señalar alguno de los aspectos que parecen tener las mayores debilidades. El mecanismo de acceso de dos factores presenta un alto grado de seguridad frente a los recursos empleados para su implementación. Obviamente, al estar basado en algoritmos como *HMAC-SHA-1* o procesos de *truncado*, las debilidades de éstos se trasladan al algoritmo final. Sin embargo, la limitación del intervalo temporal en la que es válida un OTP dificulta enormemente aprovecharse de las debilidades mencionadas porque un número significativo de ellas están basadas en fuerza bruta que, difícilmente, pueden llegar a resultados satisfactorios en los 30 segundos que tiene de validez la clave generada.

Por ello, los ataques suelen ir más dirigidos a aquellos procesos que rodean la puesta en marcha y mantenimiento del sistema. Entre ellos, quizás el más importante es el de la inicialización en el que la clave secreta compartida debe ser transportada desde el servidor hasta el lugar seguro del cliente. En este proceso, es conveniente emplear algún canal separado del habitual para el acceso.

Un ejemplo del proceso de transporte es el caso del *Google Authenticator* [39]. La clave secreta se muestra al usuario en forma de un código QR en el servidor web desde el que accede el usuario en la fase inicial. La idea es que la aplicación móvil de autenticación de Google extrae la clave del código QR. El problema es que el contenido del código QR está en texto plano. Ello significa que el atacante dirigiría su ataque al PC donde se va a presentar dicho código. Algún tipo de malware permitiría fácilmente la captura de la clave secreta, por ejemplo, mediante una captura de pantalla.

Obviamente, esto también implica que el atacante debe conocer el par *user-password* de acceso de la primera fase. Y en este caso concreto, si se conoce el usuario y contraseña se puede deshabilitar el proceso de dos fases, si se realiza antes de los 30 segundos.

Un segundo aspecto que debería cuidarse en la implementación del TOTP es el chequeo de la *repetición* del OTP. Un atacante que ha adquirido, por cualquier método, un OTP válido podría intentar entrar en el servidor dentro del intervalo de los 30 segundos, aunque la víctima ya se hubiera autenticado con el mismo OTP. Una implementación real debe impedir esta posibilidad.

Otro mecanismo que podría emplearse es el uso de *malware* en el dispositivo móvil que contiene la aplicación de transferencia de la clave compartida. Por ejemplo, desde la versión de Android 5.0, las aplicaciones pueden acceder (sin ser "*root*") a la pantalla y salvar los resultados como un fichero de una imagen. Un *malware* en segundo plano puede realizar capturas de pantalla en los momentos del intercambio y emplear mecanismos de OCR *(Optical Character Recognition)* para extraer la información deseada y enviarla a un *C&C*[1] *server (Command and Control Server).*

No obstante, todos los mecanismos, incluidos éste último, además de necesitar la obtención de acceso ilegal a los sistemas donde se llevan a cabo las tareas de inicialización y autenticación, tienen la dificultad del reducido tiempo de validez. Es decir, los tiempos de extracción de la información, envío a algún lugar centralizado para su empleo y el tiempo acceso, puede superar el umbral de los 30 segundos.

Finalmente, conviene señalar que la mayor parte de los ataques con éxito provienen de los errores del propio usuario mediante técnicas de engaño y éstas son prácticamente inevitables tecnológicamente.

[1] Un *C&C server* es un computador centralizado que se emplea para enviar comandos (normalmente maliciosos) a un *Bonet* o red de computadores comprometidos.

4 Conclusiones y líneas futuras

4.1 Conclusiones

Se ha analizado la problemática de la identificación en el acceso a servidores, basada únicamente en el empleo del par *login-password* por las grandes posibilidades de su obtención ilegal mediante el empleo de fuerza bruta de computación o de mecanismos de aceleración para su determinación.

Se han analizado soluciones basadas en la autenticación de dos factores, donde uno de ellos es el correspondiente al de contraseña estática habitual y el otro el empleo de palabras clave de un solo uso (OTP). Se ha implementado una solución local de este tipo de contraseña dinámica basada en el mecanismo criptográfico HMAC, utilizando la función hash criptográfica *SHA1* y empleando como factor móvil el tiempo actual, esto es, un sistema TOTP.

A continuación, desplegando un servidor, la solución local se ha migrado a una estructura cliente-servidor, empleando HTTP REST para la comunicación entre cliente y servidor. Posteriormente, la flexibilidad del sistema ha sido mejorada ofreciendo la movilidad en el cálculo y la validación empleando una aplicación móvil implementada sobre el sistema operativo Android.

Finalmente, sin sacrificar la flexibilidad, se ha incrementado la seguridad implementando la solución sobre una tarjeta inteligente sin contactos. De esta manera, la comunicación con el servidor se lleva a cabo por la aplicación móvil, pero el procesado necesario para obtención de la palabra clave se realiza en el interior de la tarjeta por lo que el secreto compartido nunca sale al exterior.

Se ha desarrollado un ejemplo de uso de todas las aplicaciones en los distintos dispositivos y se ha comprobado experimentalmente su correcto funcionamiento. Aunque la distancia hasta un sistema real es elevada, se ha podido comprobar su operatividad en condiciones razonablemente similares a las existentes en un entorno real.

De todo el proceso se han adquirido conocimientos sobre el manejo y estructura interna de las tarjetas inteligentes, superando los distintos problemas, como el del tiempo de procesado de la tarjeta, que han ido apareciendo a lo largo del desarrollo. Asimismo, el trabajo ha permitido mejorar los conocimientos prácticos del lenguaje de comunicación Java, el entorno Android y la comunicación empleando tecnologías NFC.

Por todo ello, el sistema desarrollado se puede considerar una solución amigable y con un posible futuro dentro de los sistemas de autenticación de dos factores con tarjetas inteligentes.

El proyecto ha aunado las partes de especificación, diseño, implementación y validación de la solución, cubriendo, excepto el de los costes, todas las etapas de un proyecto de ingeniería.

4.2 Líneas Futuras

Las posibles líneas futuras del presente trabajo serían fundamentalmente aquellas que cubriesen las carencias y posibles vulnerabilidades del sistema desarrollado hasta ahora. En primer lugar, sería necesario el desarrollo de una base de datos para poder almacenar la información referida a cada uno de los clientes del sistema. Hasta el momento se ha considerado la suposición, poco realista, de la existencia de un único cliente en el sistema.

Sería necesario el establecimiento de algún canal seguro para el intercambio de la llave secreta entre el servidor y cada uno de los clientes. Cada cliente deberá tener una llave distinta, conocida por el servidor y por el propio cliente que almacenará en su tarjeta inteligente. Una posibilidad sería establecer un canal seguro para actualizar las claves de la tarjeta de forma segura extremo a extremo, haciendo que la clave viaje totalmente cifrada entre el servidor remoto y la tarjeta.

Un tercer aspecto que podría ser mejorado es el del tiempo de generación del HMAC por parte de la tarjeta inteligente, que actualmente está por encima de los 2700 ms. en el modelo disponible. La mejor solución sería encontrar una tarjeta que realice el procesado del *SHA1*, directamente, con el propio hardware de la tarjeta.

Finalmente, dado que el objetivo es mejorar la seguridad de acceso, debería hacerse un análisis sobre la fortaleza de cada uno de las aplicaciones en los distintos dispositivos. En particular, el intercambio de información entre el dispositivo móvil y el servidor, empleando la pila TCP, es uno de los lugares más susceptibles de un ataque.

Anexo I: HMAC: Keyed-Hashing for Message Authentication

Un código de autenticación en clave-*hash* (HMAC) es un mecanismo para la autenticación de mensajes empleando funciones *hash* criptográficas iterativas [10]. Típicamente, los códigos de autenticación de mensajes son utilizados entre dos partes con el objetivo de validar el intercambio de información entre ellos mediante llave secreta (*secret key*) compartida. HMAC se puede emplear en combinación con cualquier función *hash* criptográfica iterativa como puede ser *MD5* y *SHA1* en conjunción con la llave secreta compartida. La fortaleza criptográfica de HMAC depende de las propiedades de la función *hash* subyacente.

Para calcular HMAC sobre un conjunto de datos que denominaremos *"text"*, se realizan las siguientes acciones:

$$H(K\ XOR\ opad, H(K\ XOR\ ipad, text)) \quad (1)$$

Donde, *H* es una función *hash* criptográfica aplicada sobre bloques de datos de longitud *B* bytes (típicamente 64). La salida será un bloque de longitud *L* bytes (típicamente *L*=20 bytes para *SHA1* y L=16 para MD5). La llave *K* puede tener cualquier longitud menor o igual que *B* y debe ser mayor que *L*. Por último, *ipad* y *opad* son dos *strings* definidos como el byte 0x36 repetido *B* veces (*ipad*) y el 0x5C repetido *B* veces (*opad*).

Las acciones que se llevan a cabo en la ecuación (1) son las siguientes:

1) Se añaden ceros al final de *K* para crear un *string* de *B* bytes (por ejemplo, si *K* tiene longitud 20 y *B*=64, se le añaden 44 bytes con el valor 0)
2) Se hace la operación *bitwaise exclusive-or* (XOR) del *string B* del paso 1, con *ipad*.
3) Se añade el *stream* de datos "text" al string de B bytes, resultante del paso 2.
4) Se aplica H al *stream* generado en el paso 3.
5) Se hace la operación XOR al string de B bytes, realizado en el paso 1, con *opad*.
6) Se añade el resultado H, del paso 4, al string de B bytes del paso 5.
7) Se aplica H al *stream* generado en el paso 6 y se obtiene el resultado.

En la Figura 28, se muestra un diagrama que representa las operaciones básicas mencionadas anteriormente.

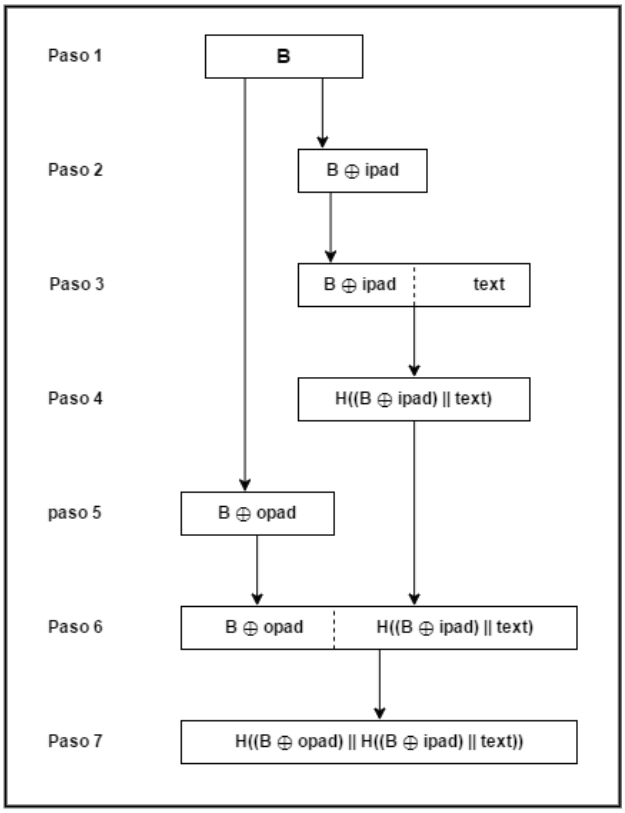

Figura 28. Construcción del HMAC

Las llaves *K* para HMAC pueden ser de cualquier longitud, aunque cualquier llave con una longitud menor de *L* bytes, no es aconsejable ya que disminuye la seguridad de la función *H*. Las llaves necesitan ser elegidas aleatoriamente y deben ser actualizadas de manera periódica.

Anexo II: Herramientas hardware y software empleadas.

A continuación, en las Tablas 15 y 16, se muestran las principales herramientas y utilidades empleadas para el desarrollo del sistema descrito en este trabajo.

Tabla 15. Software utilizado para la implementación del sistema.

Software	***Versión***	***Desarrollador***
Eclipse Java EE	Mars.2 (4.5.2)	Eclipse Foundation
Apache Tomcat	8.0	Apache Software Foundation
Android Studio	2.2	Google Inc.
GlobalPlatformPro	0.3.9	Martin Paljak y GlobalPlatform
SpringCard Scriptor XV	15.4	SpringCard SAS

Tabla 16. Hardware utilizado para la implementación del sistema.

Hardware	***Fabricante***	***Modelo***
Lector de tarjetas	Gemalto	Prox-DU Contact & Contactless
Tarjeta	Gemalto	Optelio Contactless R7
Móvil	LG	G2 D802 Versión de Android: 5.0.2

Bibliografía

[1] D. M. Turner, «Digital Authentication - The Basics,» 01 Agosto 2016. [En línea]. Available: https://www.cryptomathic.com/news-events/blog/digital-authentication-the-basics. [Último acceso: 20 Diciembre 2016].

[2] El Pais, «Dropbox reconoce el 'hackeo' de 60 millones de cuentas.,» 31 Agosto 2016. [En línea]. Available: http://tecnologia.elpais.com/tecnologia/2016/08/31/actualidad/1472642567_500051.html. [Último acceso: 15 Enero 2017].

[3] Forbes, «iCloud Data Breach: Hacking And Celebrity Photos,» 02 Septiembre 2014. [En línea]. Available: http://www.forbes.com/sites/davelewis/2014/09/02/icloud-data-breach-hacking-and-nude-celebrity-photos/#1109b96f3f69. [Último acceso: Noviembre 2016].

[4] T. V. Russell Brandom, «Your phone's biggest vulnerability is your fingerprint.,» 02 Mayo 2016. [En línea]. Available: http://www.theverge.com/2016/5/2/11540962/iphone-samsung-fingerprint-duplicate-hack-security. [Último acceso: Noviembre 2016].

[5] N. Haller, C. Metz, P. Nesser y M. Straw, «RFC 2289 - A One-Time Password System,» February 1998.

[6] G DATA Software AG, «Mobile Malware Report,» 2015. [En línea]. Available: https://public.gdatasoftware.com/Presse/Publikationen/Malware_Reports/G_DATA_Mobile MWR_Q1_2015_US.pdf. [Último acceso: 2016 Diciembre 15].

[7] Persistence Market Research (PMR), «Global Market Study on Smart Cards: Established Players Are Focusing on Smart Card Products For Telecom Application In Emerging Economies to Gain Better Market Access over Forecast Period 2016 - 2024,» 23 Junio 2016. [En línea]. Available: http://www.persistencemarketresearch.com/market-research/smart-cards-market.asp. [Último acceso: Enero 2017].

[8] Yubico, «The YubiKey Manual,» 27 Marzo 2015. [En línea]. Available: https://www.yubico.com/wp-content/uploads/2015/03/YubiKeyManual_v3.4.pdf. [Último acceso: Octubre 2016].

[9] Facebook Inc., «Whatsapp,» [En línea]. Available: https://www.whatsapp.com/.

[10] Internet Engineering Task Force (IETF), «RFC 2104 - HMAC: Keyed-Hashing for Message Authentication,» 1997.

[11] D. M'Raihi, M. Bellare, F. Hoornaert, D. Naccache y O. Ranen, «HOTP: An HMAC-Based One-Time Password Algorithm,» *RFC 4226, DOI 10.17487/RFC4226, <http://www.rfc-editor.org/info/rfc4226>.*, December 2005.

[12] M'Raihi, D., S. Machani, M. Pei y J. Rydell, «TOTP: Time-Based One-Time Password Algorithm,» *RFC 6238, DOI 10.17487/RFC6238, <http://www.rfc-*

editor.org/info/rfc6238>, May 2011.

[13] W. E. Wolfgang Rankl, Smart Card Handbook, 4th Edition, Wiley, 2010.

[14] International Organization for Standardization and International Electrotechnical Commission, ISO/IEC 7816-2 Integrated circuit cards - Part 2: Cards with contacts - Dimensions and location of the contacts, 2006.

[15] International Organization for Standardization and International Electrotechnical Commission, ISO/IEC 7816-4 Integrated circuit cards - Part 4: Organization, security and commands for interchange, 2013.

[16] International Organization for Standardization and International Electrotechnical Commission, «ISO 7816 Part 3: Electronic Signals and Transmission Protocols,» 2006.

[17] PC/SC Workgroup, «Personal Computer/Smart Card,» [En línea]. Available: https://www.pcscworkgroup.com/. [Último acceso: Diciembre 2016].

[18] Z. Chen, "Java Card Technology for Smart Cards, Architecture and Programmer's Guide", Addison Wesley, 2000.

[19] GlobalPlatform, «GlobalPlatofrm,» [En línea]. Available: https://www.globalplatform.org.

[20] International Organization for Standardization and International Electrotechnical Commission, «ISO/IEC 14443 Type A Proximity Contactless Identification Cards,» 2005.

[21] Sony Corporation, «Sony Global - FeliCa Web Site,» [En línea]. Available: http://www.sony.net/Products/felica. [Último acceso: Noviembre 2016].

[22] International Organization for Standardization and International Electrotechnical, ISO/IEC 18000-3 - Radio frequency identification for item management - Part 3: Parameters for air interface communications at 13,56 MHz, 2010.

[23] International Organization for Standardization, «ISO 18092 - Information technology - Telecommunications and information exchange between systems - Near Field Communication - Interface and Protocol (NFCIP-1),» 2004.

[24] ECMA International, «ECMA-340, Near Field Communicaction Interface and Protocol (NFCIP-1),» 2004.

[25] ETSI - European Telecommunications Standards Institute, «ETSI TS 102 190 - Near Field Communication Interface and Protocol (NFCIP-1),» 2004.

[26] International Organization for Standardization, «ISO 21481 - Information technology - Telecommunications and information exchange between systems - Near Field Communication Interface and Protocol-2 (NFCIP-2),» 2005.

[27] ECMA International, «Near Field Communication Interface and Protocol-2 (NFCIP-2)».

[28] European Telecommunications Standards Institute, «ETSI TS 102 312 - Near Field Communication Interface and Protocol-2 (NFCIP-2),» 2004.

[29] International Organization for Standardization, «ISO 15693 - Identification cards - Contactless integrated circuit cards - Vicinity cards,» 2006.

[30] Sesam Vitale, «Smart Card Alliance - Sesam Vitale Program,» [En línea]. Available: http://www.smartcardalliance.org/resources/pdf/Sesam_Vitale.pdf.

[31] Eclipse Foundation, «Eclipse Technology,» [En línea]. Available: https://eclipse.org/.

[32] G. C. &. T. W. team, «Wireshark,» [En línea]. Available: https://www.wireshark.org/. [Último acceso: Diciembre 2016].

[33] Oracle Corporation, «Java Card Platform Specification 2.2.2,» 27 Marzo 2006. [En línea]. Available: http://www.oracle.com/technetwork/java/javacard/specs-138637.html. [Último acceso: Octubre 2016].

[34] M. Paljak, «GitHub GlobalPlatformPro - Load and manage applets on compatible JavaCards,» [En línea]. Available: https://github.com/martinpaljak/GlobalPlatformPro. [Último acceso: 25 Octubre 2016].

[35] «Java Card 2.2.2 - Class Signature - javacard.security.Signature,» [En línea]. Available: https://www.win.tue.nl/pinpasjc/docs/apis/jc222/javacard/security/Signature.html. [Último acceso: Octubre 2016].

[36] «JavaCard 2.2.2 - Class MessageDigest - javacard.security.MessageDigest,» [En línea]. Available: https://www.win.tue.nl/pinpasjc/docs/apis/jc222/javacard/security/MessageDigest.html. [Último acceso: Octubre 2016].

[37] SpringCard, «SpringCard QuickStart for PC/SC,» [En línea]. Available: https://www.springcard.com/en/springcard-quickstart-for-pc-sc. [Último acceso: Agosto 2016].

[38] Google Inc., «IsoDep | Android Developers,» [En línea]. Available: https://developer.android.com/reference/android/nfc/tech/IsoDep.html. [Último acceso: 16 Agosto 2016].

[39] Google Inc., «Google Authenticator - Google Play Store Android Applications,» [En línea]. Available: https://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2. [Último acceso: 10 Octubre 2016].