

**GUÍA OPERATIVA PARA
DESARROLLADORES DE APLICACIONES**

TARJETA DNIE-DSCF 3.0

22 de agosto de 2016

Versión 1.0 Revisión 2

| | NOMBRE | FECHA |
|----------------|--|--------------|
| Elaborado por: | Área de Desarrollo – Documentos de Identificación / Tarjetas | 22/08/2016 |
| Revisado por: | | |
| Aprobado por: | | |

| HISTÓRICO DEL DOCUMENTO | | | | |
|--------------------------------|-----------------|--------------|--------------------------------|--------------|
| Versión | Revisión | Fecha | Descripción | Autor |
| 1.0 | 0 | 2/02/2015 | Creación del documento | DIT |
| 1.0 | 1 | 12/05/2015 | Actualización de ejemplos | DIT |
| 1.0 | 2 | 22/05/2016 | Optimización de la información | DIT |
| | | | | |

Índice

| | |
|--|-----------|
| ÍNDICE | 3 |
| 1 INTRODUCCIÓN | 5 |
| 1.1 Identificación del documento | 5 |
| 1.2 Propósito | 5 |
| 2 DESCRIPCIÓN DE LA TARJETA DNIE-DSCF | 6 |
| 2.1 Descripción del componente electrónico. | 7 |
| 2.2 Seguridad lógica. Mecanismos de identificación..... | 8 |
| 2.3 Ciclo de vida de la tarjeta DNIE | |
| 3 RECOMENDACIONES Y REQUISITOS DE SEGURIDAD PARA EL ENTORNO OPERACIONAL | 11 |
| 4 RELACIÓN DE COMANDOS..... | 13 |
| 4.1 Comando Get Challenge | 13 |
| 4.2 Comando Get Chip Info | 14 |
| 4.3 Comando Get Response | 15 |
| 4.4 Comando External Authenticate..... | 16 |
| 4.5 Comando Internal Authenticate | 19 |
| 4.6 Comando Manage Security Environment | 21 |
| 4.7 Comando Perform Security Operation | 28 |
| 4.8 Comando Select | 36 |
| 4.9 Comando Read Binary..... | 39 |
| 4.10 Comando Verify..... | 41 |
| 5 ESTABLECIMIENTO DE CANAL SEGURO SIN CONTACTOS..... | 45 |
| 5.1 Introducción..... | 45 |
| 5.2 Lectura del fichero Card Access | 45 |

| | | |
|------------|--|-----------|
| 5.3 | Establecimiento de un canal PACE con el algoritmo PACE-ECDH-GM-AES-CBC-CMAC-128 y utilizando como contraseña el número CAN (Impreso en el DNIE) | 47 |
| 5.3.1 | Reiniciar la tarjeta..... | 47 |
| 5.3.2 | Selección del algoritmo para PACE | 47 |
| 5.3.3 | Primer comando General Authenticate | 47 |
| 5.3.4 | Segundo comando General Authenticate..... | 48 |
| 5.3.5 | Tercer comando General Authenticate | 49 |
| 5.3.6 | Cuarto comando General Authenticate | 50 |
| 5.4 | Establecimiento de un canal PACE con el algoritmo PACE-DH-GM-AES-CBC-CMAC-128 y utilizando como contraseña el MRZ..... | 51 |
| 5.4.1 | Reiniciar la tarjeta..... | 51 |
| 5.4.2 | Selección del algoritmo para PACE | 51 |
| 5.4.3 | Primer comando General Authenticate | 52 |
| 5.4.4 | Segundo comando General Authenticate..... | 52 |
| 5.4.5 | Tercer comando General Authenticate | 54 |
| 5.4.6 | Cuarto comando General Authenticate | 55 |
| 6 | ESTABLECIMIENTO DE CANAL SEGURO | 57 |
| 7 | EJEMPLO DE ESTABLECIMIENTO DE CANAL SEGURO DE USUARIO.. | 58 |
| 7.1 | Datos externos a la tarjeta | 58 |
| 7.2 | Establecimiento de canal seguro de usuario..... | 59 |
| 7.2.1 | Reiniciar la tarjeta..... | 60 |
| 7.2.2 | Petición del número de serie de la tarjeta: | 60 |
| 8 | EJEMPLO DE ESTABLECIMIENTO DE CANAL SEGURO DE PIN Y PRESENTACIÓN DEL PIN | 77 |
| 9 | EJEMPLO DE FIRMA CON EL DNIE 3.0 | 79 |
| 9.1 | Establecimiento del canal de PIN y presentación del PIN | 79 |
| 9.2 | Establecimiento del canal de Usuario | 79 |
| 9.3 | Proceso de firma de datos | 79 |
| 10 | ACRÓNIMOS | 90 |
| 11 | BIBLIOGRAFÍA | 92 |

1 Introducción

1.1 Identificación del documento

Título: Guía operativa para desarrolladores de aplicaciones. Tarjeta DNIe-DSCF 3.0

Nombre fichero: Guía Operativa para desarrolladores.doc

Versión: 1.0

Revisión: 2

Autor: FNMT - Departamento de Documentos de Identificación – Tarjetas

Fecha: 22 de agosto de 2015

1.2 Propósito

El propósito de esta guía operativa es describir las operaciones y funcionalidades que permite la tarjeta DNIe-DSCF 3.0 a los desarrolladores de aplicaciones.

2 Descripción de la tarjeta DNIE-DSCF

La tarjeta DNIE-DSCF es una tarjeta inteligente con capacidades criptográficas configurada para ser utilizada como dispositivo seguro de creación de firma (DSCF). Sus especificaciones técnicas están basadas en normas internacionales sobre tarjetas inteligentes, así como en las recomendaciones del grupo de trabajo PC/SC.

Es una tarjeta “multi-aplicación” capaz de desarrollar su funcionalidad en diferentes entornos de operación atendiendo a sus propias funciones y políticas de seguridad.

El DNIE-DSCF está especialmente diseñado para su uso como dispositivo seguro de creación de firma y/o mecanismo de autenticación, siendo su entorno de operación más aceptado el que desarrollan las infraestructuras de clave pública.

Por tanto, se presenta como un dispositivo idóneo para aquellos escenarios en los que se requiere autenticación del usuario ante al sistema, autenticación del usuario frente a la propia tarjeta, autenticación de la entidad o componente con el que se invoca la funcionalidad de la tarjeta, integridad, confidencialidad y/o no repudio en origen.

Hay que subrayar que la tarjeta siempre mantiene el material criptográfico sensible dentro de la misma (no se permite su exportación), protegiendo su uso mediante los correspondientes mecanismos de control de acceso. De esta forma, se obtiene una considerable ventaja en términos de seguridad y portabilidad sobre las soluciones software.

La tarjeta DNIE-DSCF tiene soporte para biometría con algoritmo “Match on Card”, es decir, la verificación de los datos biométricos frente a los datos de referencia se realiza dentro de la propia tarjeta. Por tanto, se mantienen los datos sensibles de biometría siempre internos a la tarjeta, y su utilización está controlada mediante control de acceso. Esta característica de “Match on Card” confiere una importante diferencia frente a algoritmos “Match off Card”, donde la tarjeta sólo es utilizada como soporte de los datos para la verificación externa.

La funcionalidad de la tarjeta se desarrolla e implementa a través de su sistema operativo, que gestiona los recursos hardware y operaciones de bajo nivel del chip.

Otro rasgo diferenciador es la posibilidad de que las claves RSA sean generadas por el emisor y almacenadas en un estado inactivo. Así se asegura que las claves no son operativas hasta que un usuario en conocimiento de una clave de activación desencadene el proceso interno de descifrado.

El sistema operativo prevé la posibilidad de definir una estructura de ficheros acorde a la recomendación [PKCS#15] de RSA con el fin de facilitar la interoperabilidad con aplicaciones basadas en tarjeta inteligente. Esta estructura de ficheros y datos es de tipo árbol.

El DNIE-DSCF también proporciona un mecanismo fiable para el establecimiento y la gestión de un canal de comunicación seguro entre la tarjeta y el mundo exterior. El establecimiento de este canal seguro, se realiza según la norma [EN14890-1], que incluye el uso de certificados verificables por la tarjeta, para la autenticación de la entidad externa.

2.1 Descripción del componente electrónico.

El propósito del microprocesador que incorpora el DNI electrónico es contener y custodiar los datos de filiación del ciudadano, los datos biométricos (modelo dactilar, foto y firma manuscrita) y los dos pares de claves RSA con sus respectivos certificados.

La información en el chip está distribuida en zonas de seguridad con diferentes niveles y condiciones de acceso:

- Zona pública: Accesible en lectura sin restricciones.
 - Certificado de Firma.
 - Certificado de Identificación.
 - Certificado AC intermedia.
 - Claves Diffie-Hellman.
 - Certificado x509v3 de componente.
- Zona privada: El ciudadano permite su utilización mediante la introducción de la clave personal de acceso o PIN; contiene:
 - Clave privada de Firma.
 - Clave privada de Identificación.

Datos criptográficos que incorporan los DNI electrónicos.

- Parejas de claves RSA de identificación y firma.
- Clave Pública de la AC raíz.
- Claves Diffie-Hellman.

El chip almacena los siguientes certificados electrónicos:

- **Certificado de componente.** Su propósito es la identificación de la tarjeta del DNI electrónico durante el protocolo de autenticación mutua definido en [EN14890-1].

- Permite el establecimiento de un canal cifrado e identificado entre la tarjeta y el middleware.
- Su clave privada asociada no es externamente accesible.
- **Certificado de identificación.** Tiene como finalidad garantizar electrónicamente la identidad del ciudadano al realizar una transacción telemática. El certificado de identificación asegura que la comunicación electrónica se realiza con la persona que se identifica con el mismo. El titular podrá, a través de su certificado, acreditar su identidad frente a cualquiera ya que se encuentra en posesión del propio certificado y de la clave privada asociada al mismo.

El uso de este certificado no está habilitado para operaciones que exijan no repudio de origen, por tanto los terceros aceptantes y los prestadores de servicios no tendrán garantía del compromiso del titular del DNI con el contenido firmado. Su uso principal será el de generar mensajes de identificación en el seno de determinados protocolos de establecimiento de canales cifrados.

Este certificado puede ser utilizado también como medio de identificación para la realización de un registro que permita la expedición de certificados reconocidos por parte de entidades privadas, sin verse estas obligadas a realizar una fuerte inversión en el despliegue y mantenimiento de una infraestructura de registro.

- **Certificado de firma.** Destinado a la firma de documentos garantizando la integridad del documento y el no repudio de la autoría.

Es un certificado X509.v3 estándar, que tiene activo en el atributo *KeyUsage* el bit *ContentCommitment* (compromiso con el contenido) y que está asociado a una pareja de claves generada en el interior del chip del DNI electrónico.

Es este certificado, expedido por una Autoridad de Certificación reconocida, el que convierte la firma electrónica avanzada en firma electrónica reconocida, permitiendo su equiparación legal con la firma manuscrita ([Ley59/2003] y [DIR]).

2.2 Seguridad lógica. Mecanismos de identificación.

El sistema operativo DNIE dispone de distintos métodos de identificación mediante los que una entidad externa demuestra legitimidad para el uso de los datos del chip.

Usamos la expresión entidad externa para poder englobar cualquier tipo de implementación software o hardware. Dicha entidad puede ser una aplicación programada en un lenguaje de alto nivel corriendo sobre un determinado sistema operativo comercial o un programa en código máquina embebido en un periférico diseñado a medida para su ejecución.

La correcta realización de cada uno de estos métodos permite obtener unas condiciones de seguridad que podrán ser requeridas para el acceso a los distintos recursos de la tarjeta.

Identificación de usuario mediante código CHV.

La tarjeta DNIe soporta verificación de usuario (CHV-Card Holder Verification). Esta operación la realiza el sistema operativo cotejando el valor facilitado por la entidad externa con el valor almacenado en el chip. El código CHV tiene su propio contador de intentos. Tras una presentación válida el contador de intentos es automáticamente puesto a su valor inicial. El contador de intentos es decrementado cada vez que se realiza una presentación errónea, bloqueándose la posibilidad de identificación por este método si el contador llega a cero.

Identificación de aplicación.

El propósito de este método es que la entidad externa demuestre tener conocimiento de una clave como condición de acceso a determinados recursos presentes en el chip. El método elegido es un protocolo de desafío-respuesta, con los siguientes pasos:

- La aplicación pide un desafío a la tarjeta.
- Sobre ese desafío, la entidad externa aplica un algoritmo en el que participa la clave privada de un certificado conocido por el sistema operativo del DNIe. El resultado lo transmite al chip.
- El sistema operativo del chip realiza las operaciones necesarias para determinar si la clave privada es congruente con el certificado confiable con el que cuenta. En caso de coincidir, considera correcta la presentación para posteriores operaciones.

Identificación mutua.

Este procedimiento permite que cada una de las partes (tarjeta y aplicación externa) confíe en la otra, mediante la presentación mutua de certificados y su verificación.

Este proceso, similar al anterior en líneas generales, tiene como objetivo no sólo identificar los elementos participantes sino también un intercambio de claves de sesión, que deberán ser utilizadas para cifrar todos los mensajes que se intercambien posteriormente. Este servicio permite el uso de diferentes alternativas, que podrán seleccionarse implícitamente en función de la secuencia de comandos, o explícitamente, indicando su identificador de algoritmo en un comando de gestión de entorno de seguridad anterior (MSE).

Las dos opciones disponibles están basadas en la especificación [EN14890-1] y son las siguientes:

- Autenticación con intercambio de claves (capítulo 8.4 de [EN14890-1])

- Autenticación de dispositivos con protección de la privacidad, (capítulo 8.5 de [EN14890-1])

Protección de mensajes.

La sistema operativo DNIe incorpora la posibilidad de establecer un canal seguro entre el terminal y el chip que proteja los mensajes transmitidos, de hecho el sistema operativo rechazará la ejecución de determinados comandos si no se envían a la tarjeta a través de un canal seguro e identificado. Para el establecimiento es necesaria la autenticación previa del terminal y el chip mediante el uso de certificados. Durante la presencia del canal seguro cada mensaje se cifra y autentica de forma que se asegura una comunicación “uno a uno” entre los extremos del canal. El canal seguro puede ser requerido por la aplicación o puede ser una restricción de acceso impuesta por algún recurso de la tarjeta.

Para el establecimiento del canal seguro, en primer lugar se realiza un intercambio de las claves públicas de la tarjeta y el terminal mediante certificados que serán verificados por ambas partes. A continuación se ejecuta un protocolo dirigido al establecimiento de una clave de sesión. El procedimiento descrito es una variante del protocolo Diffie-Hellman.

Una vez concluido el protocolo para el establecimiento de la clave de sesión todos los mensajes deben transmitirse cifrados con esta clave.

Funcionalidades criptográficas.

- Claves RSA

El sistema operativo DNIe es capaz de generar y gestionar claves RSA. La generación de la pareja de claves RSA sigue el estándar [PKCS#1]. Se usa el algoritmo Miller-Rabin como test de primalidad.

- Firmas electrónicas

El DNIe tiene capacidad para la realización de firmas electrónicas de dos modos diferentes:

- Modo raw.
- Modo relleno PKCS#1.

3 Recomendaciones y requisitos de seguridad para el entorno operacional

Los requisitos de seguridad, tanto para el DNIE como para su entorno de operación están descritos en [PP] y en [STDNIE].

De ésta última, obtenemos que para asegurar la confidencialidad y la integridad de los activos gestionados por el DNIE, su entorno de operación debe cumplir los siguientes requisitos.

- **OE.SVD_Auth:** La aplicación de generación de certificados debe verificar la autenticidad de la clave pública.

La aplicación de generación de certificados debe verificar que el DNIE es el remitente en la exportación de la clave pública recibida, así como la integridad de la misma. La aplicación de generación de certificados debe verificar la correspondencia entre la clave privada en el DNIE del ciudadano y la clave pública en el certificado reconocido.

- **OE.CGA_Qcert:** La aplicación de generación de certificados debe generar certificados reconocidos que incluyan:

- a) El nombre del ciudadano titular del DNIE,
- b) La clave pública que se corresponde con la clave privada implementada en el DNIE bajo el único control del ciudadano, y
- c) La firma electrónica reconocida de la DGP.

La aplicación de generación de certificados debe confirmar con el certificado reconocido generado que la clave privada correspondiente a la clave pública está almacenada en el dispositivo seguro de creación de firma.

- **OE.SSCD_Prov_Service:** El DNIE debe ser auténtico.

La DGP debe gestionar la autenticidad del DNIE para ser preparado con el usuario legítimo como firmante; también personaliza y entrega el DNIE como dispositivo seguro de creación de firma al ciudadano.

- **OE.HID_TC_VAD_Exp:** Protección de los datos de verificación de autenticación (PIN, Huella dactilar).

El dispositivo externo donde se autentique el ciudadano debe garantizar la confidencialidad y la integridad de los datos de verificación de autenticación (PIN, Huella dactilar) según lo necesite el método de autenticación empleado.

Recomendaciones y requisitos de seguridad para el entorno operacional

- **OE.DTBS_Intend:** La aplicación de creación de firma envía los datos que se pretenden firmar.

El ciudadano usando la aplicación de creación de firma confiable, la cual genera el “hash” que representa los datos a ser firmados que pretenden ser firmados por el ciudadano en una forma apropiada para la firma por el DNIe, envía el “hash” al DNIe habilitando la verificación de la integridad del “hash” por el DNIe. La firma producida por el DNIe se adjunta a los datos a ser firmados o se proporciona de manera separada.

- **OE.SCA_TC_DTBS_Exp:** La aplicación de creación de firma debe proteger los datos a ser firmados.

El entorno operacional debe asegurar que el “hash” no puede ser alterado en el tránsito entre la aplicación de creación de firma y el DNIe.

- **OE.Signatory:** Obligación de seguridad del ciudadano.

El ciudadano debe revisar que las claves privadas almacenadas en el DNIe proporcionado por la DGP están en estado no operacional. El ciudadano debe mantener la confidencialidad de su PIN.

- La clave RSA de firma e identidad se deberá generar con un tamaño de clave de 2048 bits.

4 Relación de comandos

4.1 Comando Get Challenge

Definición del comando

El comando “Get Challenge” genera la emisión de un desafío (p.e. un número aleatorio) que podrá formar parte de un procedimiento de seguridad del tipo desafío-respuesta (p.e. Comando “External Authenticate”).

Estructura del comando

| Byte | Valor | Significado |
|-------|-------|---|
| CLA | 0x00 | |
| INS | 0x84 | |
| P1 | 0x00 | |
| P2 | 0x00 | |
| LC | | Vacío |
| Datos | | Vacío |
| LE | 0x08 | Longitud del desafío de 8 bytes, para establecimiento de canal seguro |



Mensaje de respuesta

| | Significado |
|----------------|-------------------|
| Campo de datos | Datos del desafío |
| SW1-SW2 | Bytes de estado |

Condiciones de estado

| SW1 | SW2 | Significado |
|------|------|------------------------------|
| 0x67 | 0x00 | Longitud incorrecta |
| 0x6A | 0x86 | Parámetros incorrectos P1-P2 |

Ejemplo

| Terminal | Sentido de la comunicación | Tarjeta |
|----------------|---|---|
| 00 84 00 00 08 |  | |
| |  | xx xx xx xx xx xx xx xx 90 00 (Número aleatorio, 8 bytes) |

4.2 Comando Get Chip Info

Definición del comando

El comando “Get chip info” permite obtener el número de serie del chip.

Estructura del comando

| Byte | Valor | Significado |
|------|-------|-------------|
| CLA | 0x90 | |
| INS | 0xB8 | |
| P1 | 0x00 | |
| P2 | 0x00 | |
| LC | | Vacío |
| Data | | Vacío |
| LE | | 0x07 |



Mensaje de respuesta

| Long. | Significado |
|---------|-----------------|
| 0x07 | Número de serie |
| SW1-SW2 | Bytes de estado |

Condiciones de estado

| SW1 | SW2 | Significado |
|------|------|--|
| 0x67 | 0x00 | Longitud incorrecta (el campo Lc no es correcto) |
| 0x6A | 0x86 | Parámetros P1-P2 incorrectos |

Ejemplo

| Terminal | Sentido de la comunicación | Tarjeta |
|----------------|---|---|
| 90 B8 00 00 07 |  | |
| |  | xx xx xx xx xx xx xx 90 00 (Número de serie, 7 bytes) |

4.3 Comando Get Response

Descripción

El comando “Get Response” es empleado para transmitir desde la tarjeta al dispositivo interfaz la respuesta APDU (o parte) que de otra manera no podría ser transmitida por el protocolo T=0.

Estructura del comando

| Byte | Valor | Significado |
|------|-------|--|
| CLA | 0x00 | |
| INS | 0xC0 | |
| P1 | 0x00 | |
| P2 | 0x00 | |
| LC | | Vacío |
| Data | | Vacío |
| LE | | Longitud de los datos esperados como respuesta |



Mensaje de respuesta

| | Significado |
|----------------|--|
| Campo de datos | Mensaje respuesta(o parte) de acuerdo a LE |
| SW1-SW2 | Bytes de estado |

Condiciones de estado

| SW1 | SW2 | Significado |
|------|------|---|
| 0x61 | 0xXX | Ejecución correcta. Quedan datos disponibles en la tarjeta. |
| 0x69 | 0x85 | Condiciones de uso no satisfechas |
| 0x6C | 0xXX | Longitud incorrecta (XX indica el valor correcto) |
| 0x90 | 0x00 | Ejecución correcta. |

Ejemplo

| Terminal | Sentido de la comunicación | Tarjeta |
|---|---|---|
| 00 C0 00 00 xx Donde xx es la longitud de datos solicitados, en este ejemplo: 0A h |  | |
| |  | xx xx xx xx xx xx xx xx xx xx 90 00 (Datos comando, 10 bytes) |

4.4 Comando External Authenticate

Definición del comando

El comando “External Authenticate” completa la ejecución de un protocolo de tipo desafío-respuesta con el fin de autenticar el terminal como parte un intercambio de claves para el establecimiento de canal seguro según el apartado 8.4 de [EN14890-1]. Está basado en claves RSA de 1024 bits y requiere que la clave pública del terminal sea cargada en la tarjeta mediante un certificado.

Estructura del comando

| Byte | Valor | Significado |
|-------|-------|--|
| CLA | 0x00 | |
| INS | 0x82 | |
| P1 | 0x00 | |
| P2 | 0x0X | Key Id |
| LC | | Longitud de los datos, coincide Long Kpub de la tarjeta. |
| Datos | | Datos |
| LE | | Vacio |

E[PK.ICC.AUT](SIGMIN)

Donde

SIGMIN = min (**SIG**, N.IFD - **SIG**)

y

SIG= DS[SK.IFD.AUT]

(

‘6A’ = relleno según [ISO9796-2] (DS esquema 1)

PRND2 = ‘XX ... XX’ bytes de relleno aleatorios generados por el terminal. La longitud debe ser la necesaria para que la longitud desde ‘0x6A’ hasta ‘0xBC’ coincida con la longitud de la clave RSA

KIFD = Semilla de 32 bytes, generada por el terminal, para la derivación de claves del canal seguro.

h[PRND2 || KIFD || RND.ICC || SN.ICC] = hash SHA-256 que incluye los datos aportados por la tarjeta y por el terminal

‘BC’ = relleno según [ISO9796-2] (opción 1)

)

El bloque de datos ‘0x6A’ || PRND2 || K_{IFD} || h || ‘0xBC’ es firmado con la clave privada del terminal (SK.IFD.AUT).

Al resultado de esta firma (**SIG**) se le aplica la función SIGMIN para luego poder encriptar el resultado con la clave pública de la tarjeta (PK.ICC.AUT).

Para poder realizar esta operación es necesario que el tamaño de las dos claves RSA implicadas (PK.ICC.AUT y SK.IFD.AUT) sea el mismo.

La tarjeta descryptará el mensaje utilizando su clave privada y verificará la firma utilizando la clave pública del terminal, que deberá haber sido cargada previamente mediante un certificado verificable en la tarjeta. Ambas claves deben estar seleccionadas previamente en la plantilla de autenticación.

Mensaje de respuesta

| | Significado |
|----------------|-----------------|
| Campo de datos | Vacío |
| SW1-SW2 | Bytes de estado |

Condiciones de estado

| SW1 | SW2 | Significado |
|------|------|--|
| 0x63 | 0xCX | El código de autenticación no es válido. X expresa el número de intentos posibles. |
| 0x65 | 0x81 | Error de memoria |
| 0x67 | 0x00 | Longitud incorrecta |
| 0x69 | 0x83 | Método de autenticación bloqueado |
| 0x69 | 0x85 | Condiciones de uso no satisfechas |
| 0x6A | 0x80 | Error en el campo de datos |
| 0x6A | 0x86 | Parámetros incorrectos P1-P2 |
| 0x6A | 0x87 | Lc inconsistente con P1-P2 |
| 0x6A | 0x88 | Datos referenciados no encontrados |

Ejemplo

| Terminal | Sentido de la comunicación | Tarjeta |
|---|---|---------|
| 00 82 00 00 P3 xx xx xx xx xx donde: P3: long clave pública de tarjeta xx xx xx xx xx: Datos para autenticación externa por parte del terminal |  | |



4.5 Comando Internal Authenticate

Definición del comando

El comando “Internal Authenticate” inicia el cálculo de una autenticación del origen de la tarjeta. El sistema de transmisión manda un desafío a la tarjeta que se firma con la clave privada RSA interna de identificación. Este comando forma parte del esquema de establecimiento de canal seguro con intercambio de claves según el apartado 8.4 de [EN14890-1]. Se utilizan claves RSA de hasta 2048 y una firma SHA256.

Estructura del comando

| Byte | Valor | Significado |
|-------|-------|--|
| CLA | 0x00 | |
| INS | 0x88 | |
| P1 | 0x00 | Autenticación interna con establecimiento de canal |
| P2 | 0x00 | Autenticación según [EN14890-1] |
| LC | 0x10 | RND.IFD SN.IFD Autenticación según [EN14890-1] ap. 8.4 |
| Datos | | RND.IFD SN.IFD Autenticación según [EN14890] ap. 8.4 |
| LE | | Vacío |

Mensaje de respuesta

Autenticación según el apartado 8.4 de [EN14890-1]

| |
|---|
| <p>$E[PK.IFD.AUT]$ (SIGMIN)</p> <p>Donde $SIGMIN = \min(SIG, N.ICC - SIG)$</p> <p>y</p> <p>$SIG = DS[SK.ICC.AUT]$</p> <p>(</p> <p>‘6A’ = relleno según [ISO9796-2] (DS esquema 1)</p> <p>PRND1 = ‘XX ... XX’ bytes de relleno aleatorios generados en la tarjeta.</p> |
|---|

La longitud debe ser la necesaria para que la longitud desde '0x6A' hasta '0xBC' coincida con la longitud de la clave RSA.
KICC = Semilla de 32 bytes, generada por la tarjeta, para la derivación de claves del canal seguro.
h[PRND1 || KICC || RND.IFD || SN.IFD2] = hash SHA-256 que incluye los datos aportados por la tarjeta y por el terminal
'BC' = relleno según [ISO9796-2] (opción 1)
)

El bloque de datos '0x6A' || PRND1 || K_{ICC} || h || '0xBC' es firmado con la clave privada de la tarjeta (SK.ICC.AUT), que debe estar seleccionada en la plantilla de autenticación.

Al resultado de esta firma (**SIG**) se le aplica la función SIGMIN para luego poder encriptar el resultado con la clave pública del terminal (PK.IFD.AUT), que previamente se deberá haber cargado (mediante certificados verificables en la tarjeta), y seleccionado en la plantilla de autenticación.




Para poder realizar esta operación es necesario que el tamaño de las dos claves RSA implicadas (PK.IFD.AUT y SK.ICC.AUT) sea el mismo.

Condiciones de estado

| SW1 | SW2 | Significado |
|------|------|--|
| 0x61 | 0xXX | SW2 indica el número de bytes disponibles en la tarjeta como datos de respuesta. |
| 0x62 | 0x83 | El fichero seleccionado está invalidado |
| 0x65 | 0x81 | Error de memoria |
| 0x69 | 0x82 | Condiciones de seguridad no satisfechas |
| 0x69 | 0x85 | Condiciones de uso no satisfechas |
| 0x6A | 0x80 | Error en el campo de datos |
| 0x6A | 0x82 | No se ha encontrado el EF criptográfico |
| 0x6A | 0x86 | Parámetros incorrectos P1-P2 |
| 0x6A | 0x88 | Datos referenciados no encontrados |

Ejemplo

| Terminal | Sentido de la comunicación | Tarjeta |
|-------------------------------------|---|---------|
| 00 88 00 00 10 xx .. xx yy .. yy |  | |

| | | |
|--|---|--|
| Donde xx .. xx desafío del Terminal yy .. yy n° serie Terminal | | |
| |  | 61 LL |
| 00 C0 00 00 LL |  | |
| |  | xx xx xx xx xx 90 00 (Datos de la tarjeta para la autenticación interna) P3 = longitud clave de la tarjeta |

4.6 Comando Manage Security Environment

Definición del comando

Este comando permite la selección de las diferentes claves y algoritmos que serán utilizados en operaciones posteriores de los comandos “Perform Security Operation”, “External Authentication” e “Internal Authentication”.

Para ello se utiliza el concepto de ‘entorno de seguridad’ (SE) definido en [ISO7816-4]. Estos SE incluyen diferentes plantillas, cada una de ellas con una función determinada, en las que se pueden seleccionar distintos parámetros necesarios para operaciones criptográficas posteriores.

Algunos de los parámetros incluidos en estas plantillas son referencias a distintas claves existentes en la tarjeta. Los formatos de estas referencias se describen en la siguiente tabla:

| Tipo de referencia | Tamaño | Formato | Descripción |
|--------------------|---------|---------------------|---|
| KRT_CRYPT0 | 2 bytes | 0x01 KeyID | Referencia una clave RSA privada almacenada en “ICC.CRYPTO” con el identificador “KeyID”. |
| KRT_IDENT | 2 bytes | 0x02 KeyID | Referencia una clave RSA (pública o privada) almacenada en “ICC.ID” con el identificador “KeyID”. |
| KRT_DIFFIE_HELLMAN | 2 bytes | 0x03 KeyID | Referencia una clave Diffie Hellman almacenada en “IDD.ID” con el identificador “KeyID”. |
| KRT_MEMORY | N bytes | CHR del certificado | Referencia una clave pública RSA en memoria. Su valor debe coincidir con el campo “Certificate Holder Reference” (CHR) incluido en el certificado utilizado |

| | | |
|--|--|-----------------------|
| | | para cargar la clave. |
|--|--|-----------------------|

A continuación se detallan las plantillas definidas y los parámetros permitidos por cada una:

| Plantilla | Parámetros |
|-----------------------------|--|
| Autenticación (AT) | <ul style="list-style-type: none"> Referencia a la clave privada de la tarjeta para la autenticación. Tiene que estar en "ICC.ID" y con un identificador entre "0x10" y "0x1F". Referencia a la clave pública del terminal para la autenticación. Tiene que estar en memoria y haber sido cargada mediante un certificado de terminal correctamente verificado. Identificador de algoritmo indicando el tipo de autenticación que se va a realizar. Datos auxiliares para su posterior verificación. Pueden incluir la fecha mínima de nacimiento requerida (Ej. 18 años antes de la fecha actual) y la fecha de caducidad máxima requerida (Ej. no debe estar caducado en la fecha actual). |
| Firma (DST) | <ul style="list-style-type: none"> Referencia a la clave privada con la que se quiere realizar una operación de firma (con PSO: Sign). Tiene que estar en "ICC.CRYPTO". Referencia a una clave pública que se quiere utilizar para verificar un certificado. Puede tratarse de una clave de CA raíz, alojada en "ICC.ID" (con identificador entre "0x00" y "0x0F"), o de una clave en memoria cargada mediante un certificado de autoridad certificadora correctamente verificado. Identificador de algoritmo indicando el tipo de relleno que se debe utilizar en una operación de firma posterior. Puede ser [PKCS#1] o [ISO9796-2]. |
| Intercambio de claves (KAT) | <ul style="list-style-type: none"> Referencia a los parámetros Diffie Hellman con los que se realizará un intercambio de claves. Parámetro KIFD en una negociación Diffie Hellman. |
| Hash (HT) | <ul style="list-style-type: none"> Referencia al algoritmo de resumen (hash) a emplear: SHA256. |

Estructura del comando

| Byte | Valor | Significado |
|------|-------|--|
| CLA | 0x00 | |
| INS | 0x22 | |
| P1 | 0xX1 | Tipo de operación |
| P2 | 0xXX | Plantilla a modificar |
| LC | | Vacío o longitud del campo de datos |
| Data | | 'Data Objects' a incluir en la plantilla |
| LE | | |

El parámetro P1 indica que tipo de operación se va a realizar, y que elementos se están suministrando a la plantilla. Sólo se soporta la modificación del SE actual, por lo que el segundo nibble siempre deberá ser ‘1’. La siguiente tabla muestra la codificación de este valor.

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|----|----|----|----|----|----|----|----|---|
| - | - | - | 1 | - | - | - | - | Secure messaging in command data field |
| - | - | 1 | - | - | - | - | - | Secure messaging in response data field |
| - | 1 | - | - | - | - | - | - | Computation, decipherment, internal authentication and key agreement |
| 1 | - | - | - | - | - | - | - | Verification, encipherment, external authentication and key agreement |
| - | - | - | - | 0 | 0 | 0 | 1 | SET |

La siguiente tabla muestra los posibles valores de P2, que identifican la plantilla a utilizar.

| Valor | Significado |
|-------|--------------------------------------|
| ‘A4’ | Plantilla de autenticación |
| ‘A6’ | Plantilla para intercambio de claves |
| ‘B6’ | Plantilla para firmas |

En el campo de datos se incluirán, en formato TLV, los distintos ‘Data Objects’ que se quieren seleccionar para la plantilla indicada.

Los posibles objetos de datos y su codificación se indican a continuación:

| Plantilla | DO-Tag | Significado | Codificación |
|---------------|----------------------------|--|--------------------------------|
| Autenticación | ‘83’ | Referencia de clave pública | 83 L83 ‘KeyRef de PuK.IFD.AUT’ |
| | | Uso de MRZ (PACE) | 83 01 01 |
| | | Uso de CAN (PACE) | 83 01 02 |
| | ‘84’ | Referencia de clave privada | 84 L84 ‘KeyRef de PrK.ICC.AUT’ |
| | | Uso de algoritmo MODP1024Group160 para DH (PACE) | 84 01 00 |
| | | Uso de algoritmo BrainPoolP256r1 para ECC (PACE) | 84 01 0D |
| ‘80’ | Identificador de Algoritmo | 80 L80 ‘Algorithm ID’ | |

Relación de comandos

| | | | |
|-----------------------|------|------------------------------------|--|
| | | | Los posibles valores de AlgId son: '17' RSA |
| | | OID PACE_DH_GM_AES_CBC_CMACH_128 | 80 0A 04007F00070202040102 |
| | | OID PACE_ECDH_GM_AES_CBC_CMACH_128 | 80 0A 04007F00070202040202 |
| | | Datos auxiliares de autenticación | 67 L67 'Datos auxiliares' |
| | '67' | Referencia de clave pública | 83 L83 'KeyRef' Indica la clave DH a utilizar en el intercambio de claves |
| Intercambio de claves | '83' | Semilla aleatoria | 91 L91 'Random seed' Se utiliza en el intercambio de claves DH para enviar Kifd |
| | '91' | Referencia de clave pública | 83 L83 'KeyRef de PuK.IFD.AUT' |
| Firma | '83' | Referencia de clave privada | 84 L84 'KeyRef de PrK.ICC.AUT' |
| | '84' | Identificador de Algoritmo | 80 L80 'Algorithm ID' |
| | '80' | | |

Los valores del identificador de algoritmo soportados son:

| Identificador | Algoritmos | Descripción |
|---------------|--|--|
| 'FF 04 00 00' | Hash SHA256 | Algoritmo de hash |
| 'FF 14 01 00' | Firma digital RSA PKCS#1 v1.5 con SHA256 | Firma digital |
| 'FF 44 20 4C' | Protocolo de transporte de clave RSA con SHA256 Cifrado AES MAC con AES-CMAC | Establecimiento de canal seguro con intercambio de claves según el apartado 8.4 de [EN14890-1] |

| | | |
|--|---------------------------------|--|
| | Derivación de claves con SHA256 | |
|--|---------------------------------|--|

Cuando se utiliza una plantilla de autenticación con datos auxiliares para su posterior verificación, la codificación de este objeto de datos debe ser como se indica a continuación:

| | | | | | | |
|------|-----|------|-----|-----------------------------|-----|--------------------|
| ‘67’ | L67 | | | | | |
| | | ‘73’ | L73 | Discretionary Data Template | | |
| | | | | ‘06’ | L06 | Object Identifier |
| | | | | ‘53’ | L53 | Discretionary Data |
| | | ... | | | | |

En esta selección de datos auxiliares, el ‘Object Identifier’ (OID) indica el tipo de dato que se va a verificar. Debe ser uno de los tipos de datos definidos en el

Comando Verify.

El campo ‘Discretionary Data’ debe contener el valor de referencia a utilizar en las posteriores verificaciones para el tipo de datos indicado en el OID anterior.

Tanto para el caso de verificación de fecha de nacimiento del titular, como en el de verificación de fecha de caducidad del documento, el campo deberá estar codificado en una estructura ASN1 Date definida de la siguiente forma:

Date ::= NumericString (SIZE (8)) -- YYYYMMDD

Mensaje de respuesta

| | Significado |
|----------------|-----------------|
| Campo de datos | Vacío |
| SW1-SW2 | Bytes de estado |

Condiciones de estado

| SW1 | SW2 | Significado |
|------|------|------------------------------|
| 0x62 | 0x83 | Fichero invalidado |
| 0x6A | 0x82 | Fichero no encontrado |
| 0x6A | 0x86 | Parámetros incorrectos P1-P2 |
| 0x6A | 0x87 | Longitud de datos incorrecta |



| | | |
|------|------|------------------------------------|
| 0x6A | 0x88 | Datos referenciados no encontrados |
|------|------|------------------------------------|

Ejemplo 1: Selección de la clave Root CA

| Terminal | Sentido de la comunicación | Tarjeta |
|-------------------------------|----------------------------|---------|
| 00 22 81 B6 04 83 02 02 0F | | |
| | | 90 00 |

Ejemplo 2: Selección de la clave en memoria

| Terminal | Sentido de la comunicación | Tarjeta |
|---|----------------------------|---------|
| 00 22 81 B6 0A 83 08 xx .. xx donde xx .. xx Identificador de la clave recuperada en memoria. CHR del certificado autoverificable (8 bytes) | | |
| | | 90 00 |



Ejemplo 3: Selección de la clave en memoria para Autenticación interna

| Terminal | Sentido de la comunicación | Tarjeta |
|---|----------------------------|---------|
| 00 22 C1 A4 0A 83 0C xx .. xx 84 02 02 1F donde xx .. xx Número de serie del Terminal (12 bytes) | | |
| | | 90 00 |

Ejemplo 4: Selección de la clave para autenticación

| Terminal | Sentido de la comunicación | Tarjeta |
|--|---|---------|
| 00 22 41 B6 04 83 02 01 01 donde 01 01: Identificador fichero de la clave para autenticación (2 bytes) |  | |
| |  | 90 00 |

Ejemplo 5: Selección de la clave para firma

| Terminal | Sentido de la comunicación | Tarjeta |
|--|---|---------|
| 00 22 41 B6 04 84 02 01 02 donde 01 02: Identificador fichero de la clave para firma o no repudio (2 bytes) |  | |
| |  | 90 00 |

4.7 Comando Perform Security Operation

Definición del comando

Este comando permite realizar las siguientes operaciones relacionadas con los procesos de autenticación y firma:

a) Validación de un certificado.

Este sub-comando permite la verificación de un certificado utilizando la clave pública de la autoridad certificadora raíz almacenada en la tarjeta, o una clave pública de una autoridad certificadora intermedia cargada anteriormente mediante otro certificado.

El formato de los certificados es el definido en el capítulo 14 de [EN14890-1]. Se soportan los certificados de acuerdo a las plantillas con identificadores (CPI) 3 y 4.

Los certificados con CPI=3 son utilizados para la cargar la clave pública de una autoridad certificadora intermedia. Su formato es el siguiente:

| | | | | |
|---------|---------|------------------|---|---|
| '7F 21' | '81 CE' | C_CV certificado | | |
| | | '5F 37' | '81 80' | Longitud de la firma (ejemplo para claves de 1024 bits) '6A' = Relleno según [ISO9796-2] CPI = '03' CAR = Referencia de Autoridad Certificadora CHR = Referencia portador tarjeta CHA = AID[1..6] Identificador de rol 'A0 00 00 01 67 45' Identificador de rol OID = '2B 24 03 04 02 02 04' PK_part1 = 'XX ... XX' (MSB..LSB) Hash = 'XX ...XX' 'BC' Los campos descritos no están en claro en el certificado, si no que son la entrada de la firma incluida en esta sección. |
| | | '5F 38' | '3D' | PK_part2 = 'XX ... XX' (MSB..LSB) '00 01 00 01' Incluye el resto de la clave pública, que no cupo en PK_part1 (resto del módulo y exponente público en 4 bytes) |
| | '42' | '08' | CAR Se incluye también en claro para poder identificar la autoridad que firma el certificado | |

La codificación de los campos es la siguiente:

- CPI** es el identificador de la plantilla con la que está construida el certificado (Certificate Profile Identifier). Este campo ocupa un único byte.
- CAR** es un campo de ocho bytes que identifica la autoridad certificadora que emitió el certificado. (Certification Authority Reference).

- ❑ **CHR** es el identificador del propietario del certificado (Certificate Holder Reference). En este caso, corresponde a la autoridad certificadora intermedia cuya clave pública contiene este certificado. Este campo ocupa siempre doce bytes, cuatro bytes de relleno a cero y el identificador de la autoridad certificadora.
- ❑ **CHA** indica los niveles de autorización que se conceden al propietario del certificado (Certificate Holder Authorisation). Es un campo de siete bytes, en los que los seis primeros son la parte más significativa del identificador de aplicación definido en [EN14890-1] (0xA0 0x00 0x00 0x01 0x67 0x45) y el último byte es el identificador de 'rol' del certificado.
- ❑ **OID** es un campo de siete bytes con el identificador de objeto definido en [EN14890-1]. Identifica la función resumen empleada. Su valor debe ser 2B 24 03 04 02 02 04 para SHA-256.
- ❑ **PK_part1** es la primera parte de la clave pública incluida en el certificado. La clave pública RSA está formada por el módulo concatenado con el exponente público codificado en 4 bytes. En este campo se incluirá el máximo número de bytes posible (en función del tamaño de las claves) y el resto deberá ir en claro en el campo PK_part2.
- ❑ **Hash** es el resultado de aplicar la función resumen a la concatenación de los campos CPI, CAR, CHR, CHA, OID y la clave pública completa.
- ❑ **PK_part2** es el campo con la parte de la clave pública que no se pudo incluir en PK_part1.
- ❑ El bloque de datos '0x6A' || CPI || CAR || CHR || CHA || OID || PK_part1 || Hash || '0xBC' constituye la parte a firmar del certificado y deberá tener el mismo tamaño que la clave RSA con la que está firmado el certificado.

Los certificados con CPI=4 son utilizados para la cargar la clave pública de un terminal, que será utilizada a continuación en un proceso de autenticación. Su formato es el siguiente:

| | | | | |
|---------|---------|------------------|---------|--|
| '7F 21' | '81 CD' | C_CV Certificado | | |
| | | '5F 37' | '81 80' | Longitud de la firma (ejemplo para claves de 1024 bits) '6A' = Relleno según [ISO9796-2] CPI = '04' CAR = Referencia de autoridad certificadora CHR = Referencia del propietario del certificado CHA = AID[1..6] Identificador de rol OID = '2B 24 07 02 01 04' PK_part1 = 'xx..xx' (MSB..LSB) Hash = 'xx..xx' 'BC Los campos descritos no están en claro en el certificado, si no que son la entrada de la firma incluida en esta sección. |
| | | '5F 38' | '3C' | PK_part2 = 'XX ... XX' (MSB..LSB) '00 01 00 01' Incluye el resto de la clave pública, que no cupo en PK_part1 (resto del módulo y exponente público en 4 bytes) |

| | | | | | |
|--|--|------|------|-----|--|
| | | '42' | '08' | CAR | Se incluye también en claro para poder identificar la autoridad que firma el certificado |
|--|--|------|------|-----|--|

La codificación de los campos es la siguiente:

- ❑ **CPI** es el identificador de la plantilla con que está construida el certificado (Certificate Profile Identifier). Este campo ocupa un único byte.
- ❑ **CAR** es un campo de ocho bytes, que identifica la autoridad certificadora que emitió el certificado (Certification Authority Reference).
- ❑ **CHR** es el identificador del propietario del certificado (Certificate Holder Reference), en este caso, corresponde al número de serie del terminal propietario del certificado. El campo ocupa siempre doce bytes. Si el número de serie es de menor longitud, se completará con ceros a la izquierda. Se considera que el número de serie es de, al menos, ocho bytes.
- ❑ **CHA** indica los niveles de autorización que se conceden al propietario del certificado (Certificate Holder Authorization). Es un campo de siete bytes, en los que los seis primeros son la parte más significativa del identificador de aplicación definido en [EN14890] (0xA0 0x00 0x00 0x01 0x67 0x45) y el último byte es el identificador de 'rol' del certificado.
- ❑ **OID** es un campo de seis bytes con el identificador de objeto definido en [EN14890]. Identifica la función resumen empleada. Su valor debe ser 2B 24 07 02 01 04 para SHA-256.
- ❑ **PK_part1** es la primera parte de la clave pública incluida en el certificado. La clave pública RSA está formada por el módulo concatenado con el exponente público codificado en 4 bytes. En este campo se incluirá el máximo número de bytes posible (en función del tamaño de las claves) y el resto deberá ir en claro en el campo PK_part2.
- ❑ **Hash** es el resultado de aplicar la función resumen a la concatenación de los campos CPI, CAR, CHR, CHA, OID y la clave pública completa.
- ❑ **PK_part2** es el campo con la parte de la clave pública que no se pudo incluir en PK_part1.

El bloque de datos '0x6A' || CPI || CAR || CHR || CHA || OID || PK_part1 || Hash || '0xBC' constituye la parte a firmar del certificado y deberá tener el mismo tamaño que la clave RSA con la que está firmado el certificado.

Los identificadores de rol soportados son los siguientes:

| Rol ID | Uso del certificado |
|-----------|---|
| 0010 0001 | Puede utilizarse para la verificación de firma en una cadena cruzada de certificados. |

| | |
|-----------|---|
| 0010 0010 | Puede utilizarse para la verificación de firma en una cadena de certificados. |
| 0000 0001 | Puede utilizarse para identificación del terminal en un proceso de establecimiento de canal seguro (y se podrá conseguir la condición de acceso PRO). |
| 0000 0010 | Puede utilizarse en el proceso de establecimiento de un canal seguro de usuario (NO se podrá conseguir la condición de acceso PRO). |
| 1001 XXX1 | Puede utilizarse para la verificación de datos firmados en una operación de carga de referencia de datos biométricos. |
| 1001 XX1X | Puede utilizarse para la verificación de datos firmados en una operación de carga de datos de un fichero de tipo 'FE transparente de datos firmados'. |
| 1010 XXX1 | Puede utilizarse para un proceso de verificación de edad del titular. |
| 1010 XX1X | Puede utilizarse para un proceso de verificación de caducidad de documento. |

b) Cálculo de un “hash” (para su posible firma posterior)

Este sub-comando permite realizar el cálculo de un código “hash” para firmarlo en un comando posterior. Según el valor del parámetro P2 del APDU enviado, se puede incluir los datos en crudo (sin estructura), o en forma de objetos de datos que permiten realizar la operación parcialmente fuera de la tarjeta.

Para realizar en la tarjeta únicamente el último paso del algoritmo, los datos deben tener la siguiente estructura:

| T | L | V |
|----------|----------|---|
| '90' | '1C' | 20 bytes con el valor intermedio del “hash” calculado externamente, seguido de 8 bytes con el número de bits de los que ya se ha calculado el “hash”. |
| '80' | 'xx' | Resto de bytes que deben ser incluidos en el “hash” (64 bytes máximo). |

Para realizar un “hash”, utilizando encadenamiento de comandos (command chaining de [ISO7816-4]), los datos a utilizar en las diferentes llamadas a este subcomando serían:

Primer comando

| T | L | V |
|----------|----------|----------|
| | | |

| | | |
|------|------|---|
| '90' | '1C' | 20 bytes con el valor intermedio del "hash" calculado externamente, seguido de 8 bytes con el número de bits de los que ya se ha calculado el "hash". |
|------|------|---|

Comandos intermedios (excepto el último)

| T | L | V |
|------|------|------------------------------|
| '80' | '40' | Datos a incluir en el "hash" |

Último comando

| T | L | V |
|------|------|-----------------------------------|
| '80' | 'xx' | Últimos datos a incluir en "hash" |

c) Firma de un "hash" calculado interna o externamente

Este sub-comando permite firmar un "hash" ya existente en la tarjeta, como resultado de una operación previa de cálculo de hash, o firmar unos datos directamente enviados en el comando. La firma se realiza con una clave privada RSA ubicada en "ICC.CRYPTO".

En el caso de que los datos ya estén disponibles en la tarjeta, la firma se podrá realizar con relleno según [PKCS#1], indicado en el identificador de algoritmo seleccionado en la plantilla. En caso de no haberse seleccionado ninguno, por defecto se usará [PKCS#1].

El formato de relleno para esta firma se muestra en la siguiente tabla:

| DSI according to [PKCS#1] V1.5 | | |
|--------------------------------|---|--|
| T | L | V (Example with 1024-bit modulus) |
| - | - | '00' = Start byte '01' = Block type 'FF..FF' = Padding String PS (minimum of 8 bytes) '00' = Separator DigestInfo = ASN.1-Sequence of OID and parameter) and digest (ASN.1-DO has value) { SHA1 : 30 21 30 09 06 05 2b 0e 03 02 1a 05 00 04 14 } 'xx..xx' = Hash |

En caso de que los datos a firmar se incluyan directamente en el comando, la firma se realizará con formato PKCS#1 como indica el apartado 6.3 de [EN14890-2]. Los datos no deberán exceder del 40% del tamaño de la clave RSA utilizada.

Estructura del comando

| Byte | Valor | Significado |
|------|-------|--|
| CLA | 0x00 | |
| INS | 0x2A | |
| P1 | 0xXX | Tag indicando el tipo de datos de la respuesta |
| P2 | 0xXX | Tag indicando el tipo de datos enviado |
| LC | 0x14 | Longitud de los datos |
| Data | | Datos de entrada |
| LE | | Datos de salida |

La siguiente tabla indica las posibles combinaciones de P1-P2 soportadas

| P1 | P2 | Operación |
|------|------|--|
| '00' | 'AE' | Verificación de certificado. Los datos de entrada son el certificado y no se esperan datos de respuesta. |
| '90' | '80' | Cálculo de "hash". En la entrada se entregan datos sin estructura y la respuesta es el "hash" calculado por la tarjeta. Este "hash" quedará disponible para una firma realizada en el siguiente comando. |
| '90' | 'A0' | Cálculo de "hash". En la entrada se entregan los datos con estructura permitiendo realizar el "hash" parcialmente fuera de la tarjeta. La respuesta será el "hash" calculado. |
| '9E' | '9A' | Cálculo de una firma. Si el tamaño de los datos de entrada es cero, se firmará el "hash" generado anteriormente. En caso contrario, se firmarán los datos enviados en el comando. La respuesta es la firma generada. |

Mensaje de respuesta

| | Significado |
|----------------|---|
| Campo de datos | Mensaje respuesta (o parte) de acuerdo a LE |

| | |
|---------|-----------------|
| SW1-SW2 | Bytes de estado |
|---------|-----------------|



Condiciones de estado

| SW1 | SW2 | Significado |
|------|------|--|
| 0x62 | 0x83 | Fichero invalidado |
| 0x65 | 0x81 | Error de memoria |
| 0x69 | 0x82 | Condiciones de seguridad no satisfechas |
| 0x69 | 0x85 | Condiciones de uso no satisfechas (secuencia de comandos incorrecta) |
| 0x69 | 0x86 | Tipo de fichero incorrecto |
| 0x6A | 0x80 | Campo de datos incorrecto |
| 0x6A | 0x82 | Fichero no encontrado |
| 0x6A | 0x86 | Parámetros incorrectos P1-P2 |
| 0x6A | 0x87 | Longitud de datos incorrecta |
| 0x6A | 0x88 | Datos referenciados no encontrados |



Ejemplo 1: Verificar Certificado Autoverificable CA intermedia

| Terminal | Sentido de la comunicación | Tarjeta |
|----------------|---|---|
| 00 2A 00 AE P3 |  | |
| |  | xx .. xx 90 00 donde: xx .. xx Certificado autoverificable de la CA intermedia (longitud P3) |


Ejemplo 2: Verificar Certificado Autoverificable Certificado Terminal


| Terminal | Sentido de la comunicación | Tarjeta |
|----------------|---|---|
| 00 2A 00 AE P3 |  | |
| |  | xx .. xx 90 00 donde: xx .. xx Certificado autoverificable del Terminal (longitud P3) |

Ejemplo 3: Cálculo de un hash en la tarjeta

| Terminal | Sentido de la comunicación | Tarjeta |
|---|---|---|
| 00 2A 90 80 P3 xx .. xx donde: xx son los datos sobre los que se calculará el hash. Longitud de los datos P3. |  | |
| |  | 61 14 0x14, hash disponible (longitud 20 bytes) |

Ejemplo 4: Firma de un hash con formato PKCS#1

| Terminal | Sentido de la comunicación | Tarjeta |
|---|---|---------|
| 00 2A 9E 9A 23 00 2A 9e 9A 23 30 21 30 09 06 05 2b 0e 03 02 1A 05 00 04 14 83 9b 54 3f b0 9d 20 c8 03 b4 6e 6f 8f 07 47 24 49 51 82 2f Datos = Digest Info + SHA1 |  | |

| | | |
|--|---|--|
| | | |
| |  | 61 xx Firma disponible formato PKCS#1 (longitud xx bytes, long de la clave RSA que firma) |

4.8 Comando Select

Definición del comando

Este comando permite la selección de fichero dedicado (DF) o de un fichero elemental (EF).

Estructura del comando

| Byte | Valor | Significado |
|------|-------|---|
| CLA | 0x00 | |
| INS | 0xA4 | |
| P1 | 0x00 | Selecciona DF o EF por Id (data field = id) |
| | 0x04 | Selección directa de DF por nombre. |
| P2 | 0x00 | |
| LC | | Longitud del campo de datos |
| Data | | Datos de acuerdo a P1-P2 |
| LE | | Vacío |

Mensaje de respuesta

| | Significado |
|----------------|---|
| Campo de datos | Plantilla FCI con objetos de datos conformes con [ISO7816-4/9]. |
| SW1-SW2 | Bytes de estado |

La información de control del fichero (FCI) es la cadena de datos que se recibe en respuesta a un comando "Select File".

| | | | |
|----------|----------|--|---|
| 0x6F | Length | FCI template with DOs according to ISO 7816-4/9. Length of the subsequent DOs. | |
| T | L | Value | |
| 0x84 | 0xXX | DF name. (Only for DFs) | |
| 0x85 | 0xXX | Proprietary information. Composed as follow: | |
| | | Length | Meaning |
| | | 0x01 | File descriptor byte: - 0x01 Elementary File (EF) - 0x34 Dedicated File (DF) - 0x15 Proprietary EF for Security purposes. Linear variable SIMPLE-TLV |
| | | 0x02 | File identifier |
| | | 0x02 | Number of data bytes in the file, excluding structural information |
| | | 0x05 | Security attributes, proprietary information. |
| | | 0x01 | Control Flag for Security files Security Efs (type 0x15) ---- 000-b CHV keys ---- 001-b App Keys ---- 010-b RFU ---- 011-b RFU 0000 100-b Empty RSA KEY file ---- 101-b RFU ---- 110-b Active RSA Key 1--- 1---b Key generated internally -10- 1---b RSA Key in CRT format -01- 1---b RSA Key without CRT ---1 1---b RFU ---- 111-b Inactive RSA Key -100 111-b Inactive RSA Key Type 1 -010 111-b Inactive RSA Key Type 2 -001 111-b Inactive RSA Key Type 3 |

| | | |
|--|------|--|
| | 0x02 | <p>Control bytes for RSA cryptographic files Cryptographic Efs (type 0x15) RSA KEY in CRT format ---- -b XXXX XXXXb bit length of the key in 64 base 0000 0000b ---- -b empty key 1--- ----b ---- -b q present -1-- ----b ---- -b p present --1- ----b ---- -b CRT present ---1 ----b ---- -b d mod (p-1) present ---- 1--b ---- -b d mod (q-1) present ---- -1-b ---- -b public exponent present ---- --1b ---- -b modulus present 1111 1011b ---- -b valid RSA with CRT key</p> <p>Inactive RSA KEY ---- -b XXXX XXXXb bit length of the key in 64 base 0000 0000b ---- -b empty key ---- -1--b ---- -b A8h-A9h-B8h present ---- --1-b ---- -b Aah-ABh-BAh present ---- --1b ---- -b Ach-BCh Hash present 0000 0111b ---- -b valid inactive RSA key</p> <p>RSA KEY without CRT ---- -b XXXX XXXXb bit length of the key in 64 base 0000 0000b ---- -b empty key ---- -1--b ---- -b private present ---- --1-b ---- -b public exponent present ---- --1b ---- -b modulus present 0000 0111b ---- -b valid RSA key</p> |
|--|------|--|

donde:





‘-‘ significa que el valor no tiene significado en el contexto y

‘X’ significa cualquier valor.





Condiciones de estado

| SW1 | SW2 | Significado |
|------|------|---|
| 0x61 | 0xXX | Ejecución correcta. Se indica el número de bytes disponibles en respuesta al comando como una estructura FCI. |
| 0x62 | 0x83 | Fichero seleccionado invalidado |
| 0x65 | 0x81 | Error de memoria (FCI erróneo) |
| 0x6A | 0x82 | Fichero no encontrado |
| 0x6A | 0x86 | Parámetros incorrectos P1-P2 |
| 0x6A | 0x87 | Lc inconsistente con P1-P2 |

Ejemplo 1: Selección del fichero elemental Id: 60 1F

| Terminal | Sentido de la comunicación | Tarjeta |
|----------------------|---|---|
| 00 A4 00 00 02 60 1F |  | |
| |  | 61 0E |
| 00 C0 00 00 0E |  | |
| |  | 6F 0C 85 0A 01 60 1F 03 25 00 FF FF FF FF 90 00 |

Ejemplo 2: Selección por nombre del fichero dedicado PKCS-15

| Terminal | Sentido de la comunicación | Tarjeta |
|--|---|--|
| 00 A4 04 00 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 |  | |
| |  | 61 1C |
| 00 C0 00 00 1C |  | |
| |  | 6F 1A 84 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 85 0A 38 50 15 00 0C FF FF FF FF FF 90 00 |

4.9 Comando Read Binary

Definición del comando

El comando “Read Binary” devuelve en su mensaje de respuesta el contenido (o parte) de un fichero elemental transparente.

Estructura del comando

| Byte | Valor | Significado |
|-------|-------|---|
| CLA | 0x0X | |
| INS | 0xB0 | |
| P1-P2 | | Offset del primer byte a leer desde el principio del fichero. |
| LC | | Vacío |
| Datos | | Vacío |
| LE | | Número de bytes a leer |

Si el campo Le=0, el número de bytes a leer es 256.





Mensaje de respuesta

| | Significado |
|----------------|-------------------------|
| Campo de datos | Datos leídos (LE bytes) |
| SW1-SW2 | Bytes de estado |

Condiciones de estado

| SW1 | SW2 | Significado |
|------|------|--|
| 0x62 | 0x83 | El EF o DF seleccionados están invalidados |
| 0x65 | 0x81 | Error de memoria |
| 0x69 | 0x82 | Condiciones de seguridad no satisfechas |
| 0x69 | 0x85 | Condiciones de uso no satisfechas |
| 0x69 | 0x86 | Comando no permitido (no existe un EF seleccionado) |
| 0x6A | 0x82 | Fichero no encontrado |
| 0x6B | 0x00 | Parámetros incorrectos (el offset está fuera del EF) |
| 0x6C | 0xFF | Longitud incorrecta (XX indica la longitud exacta) |

Ejemplo: Lectura Binaria

| Terminal | Sentido de la comunicación | Tarjeta |
|----------------|---|--|
| 00 B0 00 00 50 |  | |
| |  | xx ..xx 90 00 xx .. xx Contenido del fichero (primeros 0x50 bytes) longitud = 80 bytes, |
| 00 B0 00 50 40 |  | |
| |  | yy .. yy 90 00 yy .. yy Contenido del fichero a partir del offset 0x50 (longitud = 64 bytes), |

4.10 Comando Verify

Definición del comando

El comando “Verify” permite realizar las siguientes funciones:

- Comparar en la tarjeta los datos de verificación de usuario enviados desde el dispositivo interfaz con los almacenados en la tarjeta. Esta operación se suele denominar verificación de PIN o CHV.
- Verificar si el documento cumple los parámetros indicados en los datos auxiliares, en el caso de que se haya establecido el canal seguro con un terminal autorizado para ello. Los datos que se pueden comprobar son la fecha de nacimiento del titular, y la fecha de caducidad del documento.

Estructura del comando

A) Para verificación de PIN

| Byte | Valor | Significado |
|------|-------|-------------|
| CLA | 0x00 | |
| INS | 0x20 | |

| | | |
|------|------|-------------------------------------|
| P1 | 0x00 | |
| P2 | 0x00 | Datos de referencia (código CHV) |
| LC | | Vacío o longitud del campo de datos |
| Data | | Vacío o datos de verificación |
| LE | | |

Estructura del comando para verificación de datos auxiliares (fecha nacimiento / caducidad)

| Byte | Valor | Significado |
|------|-------|---|
| CLA | 0x00 | |
| INS | 0x20 | |
| P1 | 0x80 | |
| P2 | 0x00 | |
| LC | | Longitud del campo de datos |
| Data | | OID del tipo de dato auxiliares a verificar |
| LE | | |

Los identificadores de objeto OID utilizados para los datos auxiliares siguen el esquema definido en [ASM].

- bsi-de OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4) etsi(0) reserved(127) etsi-identified-organization(0) 7 }
- id-AuxiliaryData OBJECT IDENTIFIER ::= { bsi-de applications(3) mrtd(1) 4 }
- id-DateOfExpiry OBJECT IDENTIFIER ::= { id-AuxiliaryData 2 }
- id-DateOfBirth OBJECT IDENTIFIER ::= { id-AuxiliaryData 1 }

Los posibles valores del campo de datos son:

| Tipo de dato | OID | Codificación |
|---------------------------------|-----------------------|--------------------|
| Fecha de nacimiento del titular | 0.4.0.127.0.7.3.1.4.1 | 04007F000703010401 |

| | | |
|----------------------------------|-----------------------|--------------------|
| Fecha de caducidad del documento | 0.4.0.127.0.7.3.1.4.2 | 04007F000703010402 |
|----------------------------------|-----------------------|--------------------|


Mensaje de respuesta

| | Significado |
|----------------|-----------------|
| Campo de datos | Vacío |
| SW1-SW2 | Bytes de estado |


Condiciones de estado

| SW1 | SW2 | Significado |
|------|------|---|
| 0x63 | 0xCX | Verificación fallida: 'X' indica el número de intentos disponibles. |
| 0x63 | 0x00 | Verificación fallida en comprobación de datos auxiliares |
| 0x65 | 0x81 | Error de memoria |
| 0x67 | 0x00 | Longitud incorrecta |
| 0x69 | 0x82 | Condiciones de seguridad no satisfechas |
| 0x69 | 0x83 | Método de autenticación bloqueado |
| 0x69 | 0x83 | Identificador de CHV no permitido |
| 0x6A | 0x82 | No se encuentra el fichero CHV |
| 0x6A | 0x84 | No hay memoria suficiente |
| 0x6A | 0x86 | Parámetros incorrectos P1-P2 |
| 0x6A | 0x88 | Datos referenciados no encontrados |

Ejemplo: Presentación de PIN

| Terminal | Sentido de la comunicación | Tarjeta |
|---|---|---------|
| 00 20 00 00 08 xx .. xx donde xx .. xx: PIN |  | |

Relación de comandos

| | | |
|--|---|--|
| del usuario longitud en este ejemplo = 8 bytes | | |
| |  | 90 00 Verificación satisfactoria del PIN de usuario |

5 Establecimiento de canal seguro sin contactos

5.1 Introducción

Las especificaciones de ICAO, Supplemental Access Control for MRTD [SAC], describen un mecanismo de control de acceso basado en PACE (Password Authenticated Connection Establishment). PACE establece un canal securizado para el intercambio de mensajes securizados entre el terminal y el documento MRTD.

CEN Identification card systems – European Citizen Card [CEN-15480] y BSI Advanced Security Mechanism for MRTD – EAC [BSI-03110], indican la necesidad de establecer un canal securizado basado en el algoritmo PACE para poder acceder a la información contenida en un documento a través de la interfaz sin contactos. Esto protege que, a través de la interfaz sin contactos, se pueda acceder a información del documento sin el consentimiento del portador de la tarjeta.

PACE proporciona un mecanismo para generar claves de sesión fuertes independientemente de la fortaleza de la contraseña. La contraseña o información inicial puede tener una entropía baja (por ej. 6 caracteres son suficientes).

Los datos de la contraseña se pueden obtener, bien a partir de la información MRZ, accesible visualmente desde el documento; o bien a partir de un número CAN (Card Access Number) conocido por el portador del documento (información secreta) o que puede estar impreso en el documento y ser accesible visualmente.

En el caso del DNIe, al ser una tarjeta de interfaz dual, si deseamos establecer una comunicación mediante el interfaz sin contactos, tendremos que establecer un canal seguro a través de este interfaz, previo al establecimiento del canal seguro descrito en el apartado 6 Establecimiento de canal seguro.

5.2 Lectura del fichero Card Access

El terminal debe leer el fichero Card Access (File Id: 0x011C) de libre lectura para determinar los parámetros de los algoritmos soportados por la tarjeta o MRTD. El terminal puede utilizar cualquiera de los algoritmos y parámetros soportados.

A continuación se muestra la secuencia para la lectura del fichero Card Access para el documento DNIe.

- Selección del fichero Card Access.

```
00 A4 00 00 02 01 1C ->  
← 6F 0C 85 0A 01 01 1C 00 72 00 FF FF FF FF 90 00
```

Interpretación del FCI:

```
6F //Tag de FCI, File Control Information template  
0C // Long de FCI, 12 bytes
```



| | |
|-------------------|--|
| 85 | // Tag 85, Proprietary Information |
| 0A | // Long de Proprietary Information, 10 bytes |
| 01, | // Tipo de fichero: elemental transparente |
| 01 1C | // ID fichero |
| 00 72 | // Tamaño: 114 bytes |
| 00 FF FF FF FF FF | // Condiciones de Acceso: |
| AC1: 00 | Lectura Libre |
| AC2: FF | Escritura prohibida |
| AC3: FF | Invalidación prohibida |
| AC4: FF | Rehabilitación prohibida |
| AC5: FF | Administrador no habilitado |

- Lectura de los datos.

| | |
|---|--|
| 00 B0 00 00 72 -> | |
| <- 3170300d060804007f0007020202020101300f060a04007f000702020302010201 | |
| 013012060a04007f0007020204020202010202010d3012060a04007f000702020 | |
| 4020102010202010d3012060a04007f000702020401020201020201003012060a | |
| 04007f00070202040101020102020100 | |
| . | Interpretación de la información del fichero Card Access |
| 31 70 | |
| 30 0d | |
| 06 08 04 007f0007020202 | // OID de Terminal Authentication Info |
| 02 01 01 | // version protocolo v1 |
| 30 0F | |
| 06 0a 04007f00070202030201 | // OID de Chip Authentication Info id-CA- |
| ECDH-3DES-CBC-CBC | |
| 02 01 01 | // version protocolo v1 |
| 30 12 | |
| 06 0a 04007f00070202040202 | // OID de id-PACE-ECDH-GM-AES- |
| CBC-CMAC-128 | |
| 02 01 02 | // version protocolo PACE v2 |
| 02 01 0d | // param id: Brainpool256r1 |
| 3012 | |
| 06 0a 04007f00070202040201 | // OID de id-PACE-ECDH-GM-3DES- |
| CBC-CBC | |
| 02 01 02 | // version protocolo PACE v2 |
| 02 01 0d | // param id: Brainpool256r1 |
| 30 12 | |
| 06 0a 04007f00070202040102 | // OID de id-PACE-DH-GM-AES- |
| CBC-CMAC-128 | |
| 02 01 02 | // version protocolo PACE v2 |
| 02 01 00 | // param id. MODP1024Group160 |
| 30 12 | |
| 06 0a 04007f00070202040101 | // OID de id-PACE-DH-GM-3DES- |
| CBC-CBC | |

```
02 01 02 // version protocolo PACE v2
02 01 00 // param id. MODP1024Group160
```

- Se debe elegir uno de los algoritmos propuestos para PACE

```
PACE-ECDH-GM-AES-CBC-CMAC-128
PACE-ECDH-GM-3DES-CBC-CBC
OID de id-PACE-DH-GM-AES-CBC-CMAC-128
OID de id-PACE-DH-GM-3DES-CBC-CBC
```

5.3 Establecimiento de un canal PACE con el algoritmo PACE-ECDH-GM-AES-CBC-CMAC-128 y utilizando como contraseña el número CAN (Impreso en el DNIE)

5.3.1 Reiniciar la tarjeta

En este primer paso, reiniciamos la tarjeta para que la tarjeta pierda cualquier estado previo.

```
Dato inicial:
CAN: 313233343536 (ASCII)
```

5.3.2 Selección del algoritmo para PACE

Para ello utilizamos el comando 'MSE Set':

```
→ 0022c1a412800a04007f0007020204020283010284010d
← 9000

Datos del comando
80 0a 04007f00070202040202 // OID PACE-ECDH-GM-AES-
CBC-CMAC-128
83 01 02 // Password: CAN
84 01 0d // param id: Brainpool256r1
```

5.3.3 Primer comando General Authenticate

Empezamos el protocolo PACE con el primer comando General Authenticate, que nos devolverá cifrado el número aleatorio que se empleará en los cálculos:

```
→ 10860000027C00
← 7c12801039e979ea2c87254d98861b09345223b49000
```

Para descifrar la respuesta, tenemos que calcular la clave, que deriva del CAN:

```
// calcular sk = SHA-1( CAN || 00000003 );  
  
SHA-1(31323334353600000003)  
La clave son los 16 bytes MSB del hash  
?sk->591468cda83d65219cccb8560233600f
```

Y desciframos el dato devuelto por la tarjeta:

```
?nonce = 39e979ea2c87254d98861b09345223b4  
?secret = AES_Dec(?nonce,?sk);  
  
?secret->10ea7515cf362555ab77b7dce0384e89
```

5.3.4 Segundo comando General Authenticate

En este paso se establece un intercambio de clave Diffie Hellman. Para ello tanto la tarjeta como el terminal generan una clave efímera ECC con los parámetros de la curva brainpoolP256r1:

```
Clave IFD_DH1:  
pukIFDDH1->  
  936a1f95b40e4af3a2b2ef44f23109508c7f7781ef05c8f2f880d  
  e42fab9fba01d17d9ef4173b6c1e623019f6fd8080bc9df0f71bc  
  e18d46b700d05e648910ec
```

Se envía el comando General Authenticate con la clave pública generada por el terminal, y se recupera la generada por la tarjeta:

```
Se formatea el dato  
?data = '7C43814104'.?pukIFDDH1  
  
→10860000457c43814104936a1f95b40e4af3a2b2ef44f23109508c7  
  f7781ef05c8f2f880de42fab9fba01d17d9ef4173b6c1e623019f  
  6fd8080bc9df0f71bce18d46b700d05e648910ec  
←7c43824104644487064b4b2121d8c7e22b278b19143351e0a74a99e  
  98fc760ad0ac900bc27883e8ea2efcbf302dc7d6acf5d6adcd464  
  bd1885e6e64ff945e1e8b8846199f79000
```

Con la clave pública de la tarjeta y la privada del terminal se calcula el secreto 'H':

```
pukICCDH1->  
  644487064b4b2121d8c7e22b278b19143351e0a74a99e98fc760a  
  d0ac900bc27883e8ea2efcbf302dc7d6acf5d6adcd464bd1885e6  
  e64ff945e1e8b8846199f7  
  
// calcular blinding point H = PrkIFDDH1 * PukICCDH1  
?h = ecdh(?ecKey1,?ephKey); // ECDH
```



```
?h->
2d18f75c94bf3d81b56560cc93bccfe456c725e42101679a11dae
9dddfa02c7f7e108643a8bd98529773e5feb8b0a2d06a8e2ee830
906a9f90ee7f5e478ad05f
```

Y se calcula un nuevo punto base con el secreto H y el secreto recuperado en el primer comando:

```
// calcular nuevo punto G' = nonce*G + H
G' ->
a5233a384aea5cb5bcb9314d4a6a9a4c05f9db6e96a48a6d3f8a8
7b3c34fc9d16f3ad69c132c12b6ddcc8cdba4ea776c7235ca31f0
22e61703547e71e75f854a
```

5.3.5 Tercer comando General Authenticate

Con el nuevo punto base G' se calcula un nuevo par de claves efímeras, y se establece un nuevo intercambio de clave Diffie Hellman.

```
Clave IFD_DH2:
pukIFDDH2->
9b2b28fff0b25686488272fdd82ae2d04122447b9126857e3d755
184879273b69b223aa681b3c12f7044b774638fbb74d5791745d9
5d2e2d3cf688d7431076f4
```

Se envía el comando General Authenticate con la clave pública generada, y se recupera la generada por la tarjeta:

```
Se formatea el dato
?data = '7C43834104'.?pukIFDDH2

→10860000457c438341049b2b28fff0b25686488272fdd82ae2d0412
2447b9126857e3d755184879273b69b223aa681b3c12f7044b774
638fbb74d5791745d95d2e2d3cf688d7431076f4
←7c4384410470548e23a5fa339bfffbc87f2e6ded13dcbe9f41e3f74
671f2a72edbfff32278c1a73b761649a0a9ed49ef3d32c59ea38b3
111992fe24c34d27c8061ba475f1549000
```

Con la clave pública de la tarjeta y la privada del terminal se calcula el secreto k, a partir del cual se generan las claves de sesión del canal:

```
pukICCDH2->
70548e23a5fa339bfffbc87f2e6ded13dcbe9f41e3f74671f2a72
edbfff32278c1a73b761649a0a9ed49ef3d32c59ea38b3111992fe
24c34d27c8061ba475f154

// calcular k = PrkIFDDH2 * PukICCDH2
```

```
?k = ecdh(?ecKey1,?ephKey); // ECDH

?k->
  9645a8f7d0b7415e815db8d3e38d1812a011fcd3ec8a903b1860a
  bdea44dffee
```

5.3.6 Cuarto comando General Authenticate

Se validan las claves de sesión generadas en el paso anterior, por medio de un MAC que calcula el terminal y comprueba la tarjeta, la cual devolverá un segundo MAC.

```
// Calculo de las claves de sesión:

?kenc =sha1(?k.'00000001')
?kenc->a7e50cd7d5516612b8ab5de319ff5d2c //16MSB bytes

?kmac = sha1(?k.'00000002')
?kmac->89fc6e0275d78a6c8e5a91f9647b0ba7 //16MSB bytes
```

Se calcula el Mac del terminal:

```
// ?data = '7f494F06'. ?oid. '864104'.PukICCDH2;

?data->
  7f494f060a04007f0007020204020286410470548e23a5fa339bf
  ffbcb87f2e6ded13dcbe9f41e3f74671f2a72edbfff32278c1a73b7
  61649a0a9ed49ef3d32c59ea38b3111992fe24c34d27c8061ba47
  5f154

// cálculo del Mac. AES CMAC-mode
?mac = AES_mac(?data, ?kmac);
?mac->9c6ae86388c52d8b
```

Se envía el comando General Authenticate y se recupera el MAC devuelto por la tarjeta. El canal PACE está establecido. Se inicializa el contador de secuencia a ceros:

```
?data = '7C0A8508'. ?mac;
?data->7c0a85089c6ae86388c52d8b

->008600000c7c0a85089c6ae86388c52d8b
<--7c0a86085cafb173266bba259000
```

Comprobamos el MAC devuelto por la tarjeta:

```
// ?data = '7f494F06'. ?oid. '864104'.PukIFDDH2;
```



```
84 01 00 // param id: MODP1024Group160
```

5.4.3 Primer comando General Authenticate

Empezamos el protocolo PACE con el primer comando General Authenticate, que nos devolverá cifrado el número aleatorio que se empleará en los cálculos:

```
→ 10860000027C00  
← 7c1280108ba21f870eb3d31ca501ab5e91ed947b9000
```

Para descifrar la respuesta, tenemos que calcular la clave, que deriva del MRZ:

```
// Datos necesarios del MRZ  
- Doc Number + Check digit: '1234567897'  
- Date of Birth + Check digit: '7006207'  
- Expiration Date + Check digit: '1806209'  
  
MRZData = 3132333435363738393737303036323037  
31383036323039  
calcular sk = SHA-1(MRZData || 00000003 );  
  
La clave son los 16 bytes MSB del hash  
?sk->2e0998ac58c9c1273da90acc834230a8
```

Y desciframos el dato devuelto por la tarjeta:

```
?nonce = 8ba21f870eb3d31ca501ab5e91ed947b  
?secret = AES_Dec(?nonce, ?sk);  
  
?secret->c65bb3dd6f8930190afb7f5ba1f01b96
```

5.4.4 Segundo comando General Authenticate

En este paso se establece un intercambio de clave Diffie Hellman. Para ello tanto la tarjeta como el terminal generan una clave efímera DH con los parámetros MODP1024Group160:

```
Clave IFD_DH1:  
prKIFDDH1-->  
 8a7c676ca1d9aaba48f8854585a721e76645a5f34bd8659de3097  
 733b301fbf18141a245b91190fcab36c547cf2feae796ea3f2e6d  
 9c78c60fc65c8d7de8c05fc82836b5f7d1a26a1993928b8e173a0  
  d  
pukIFDDH1-->  
 84da6f519b938682d1dab1c91935468af1e9f8f7613836075b5b7  
 995a2f5dee618d11b41c5088a31f650c1faf2084e60f179ea2ba2
```

Establecimiento de canal seguro sin contactos

```
838277f1db5460c971aada74ccb002c7c255eba0a500f05a98680
6c26b3566e25ea04ddbf507aa107f4f212fea92fb7ad6eae1f2a9
992d36093b85a38716d636fc37e0fb77c4bebb34248a
```

Se envía el comando General Authenticate con la clave pública generada por el terminal, y se recupera la generada por la tarjeta:

```
Se formatea el dato
?data = '7c8183818180'.?pukIFDDH1

→10860000867c818381818084da6f519b938682d1dab1c91935468af
1e9f8f7613836075b5b7995a2f5dee618d11b41c5088a31f650c1
faf2084e60f179ea2ba2838277f1db5460c971aada74ccb002c7c
255eba0a500f05a986806c26b3566e25ea04ddbf507aa107f4f21
2fea92fb7ad6eae1f2a9992d36093b85a38716d636fc37e0fb77c
4bebb34248a
←7c818382818021efc7ad4e98bdf529cdad2766bfff317b2da6f4546
6b2c98f8bad1ac713a99581f77916f1c87d99f4154fcec7366cc9
1c12cabb37f91431f386fdb0939fa7bf3208d2de5bcfe2691647
8a0a68e61c64fff62acd78f66f299b5cd1a5a76330f7d7ddb18fd
8d126b69dec63e8feaa9786317a9ab45cd80fa54f2bb7ee599358
49000
```

Con la clave pública de la tarjeta y la privada del terminal se calcula el secreto 'H':

```
pukICCDH1->
21efc7ad4e98bdf529cdad2766bfff317b2da6f45466b2c98f8ba
d1ac713a99581f77916f1c87d99f4154fcec7366cc91c12cabb37
f91431f386fdb0939fa7bf3208d2de5bcfe26916478a0a68e61c
64fff62acd78f66f299b5cd1a5a76330f7d7ddb18fd8d126b69de
c63e8feaa9786317a9ab45cd80fa54f2bb7ee5993584

// calcular blinding point H = H = pukICCDH1^ prKIFDDH1
mod p
?h->
aed0c804765d725991da3b865f1f070f0904950a9b99cfda22adb
67a906df0af1cbe5c67617bb6a1484fe03482be5f790af8351939
43f4a19f71c5797a5cfd4c9f4d3d42ebc5040e2caae166d19eed9
46b9bb63b9643c8d4cebd5d9612212376bb35d15672c2047c3c32
e720d140410814ba54ad10980b5366afc6063cea463c
```

Y se calcula un nuevo generador con el secreto H y el secreto recuperado en el primer comando:

```
// calcular nuevo generador g' = g^nonce * H
G' ->
27d223f7c6cc75e200072f05918336f4f9f4a8923018983ac05f7
90e76423ce41fb92c2afe01c7fbaf43bd5a403b494d952353be75
7510ca4cab460e491aec47f8157f343410ee09efed3767184e164
```

```
4763402c8f5f413e94d98647cd0ca436096c934a115a60f7e03aa  
d903bfd58a1cbcd46004250b9d8a6dd3e10a2b7ab79f
```

5.4.5 Tercer comando General Authenticate

Con el nuevo generador g' se calcula un nuevo par de claves efímeras, y se establece un nuevo intercambio de clave Diffie Hellman.

```
Clave IFD_DH2:  
prKIFDDH1-->  
fe27a5b23c3b622aeb8dd7e4b77e301d6ab5470ab91139afd7670f48  
4a06f6e5652c9a4c0b56c24b53f4462302786b9d7f66a5940896d  
b08bb870986a86cf856f5ab4a99d7a690db0ec5abd7f9a382a7  
pukIFDDH2-->  
5ba2de272a6a6cdd9a199c8111de5b2cecf0c678c77fe0a26cda  
8c850c8707793f5e544e7c0f7bc92807489d7c34816e194afc646  
6729c89a68441b7e8b8d3e66964deb7bb02cd35a749435c1985a1  
9d40b997e08be23b91209376641084b00b60a13ece004c0381dba  
3e72e20fda6c80f9af1f018fc80934aab4b7f91f306b
```

Se envía el comando General Authenticate con la clave pública generada, y se recupera la generada por la tarjeta:

```
Se formatea el dato  
?data = '7C8183838180'.?pukIFDDH2  
  
→10860000867c81838381805ba2de272a6a6cdd9a199c8111de5b2ce  
cfa0c678c77fe0a26cda8c850c8707793f5e544e7c0f7bc928074  
89d7c34816e194afc6466729c89a68441b7e8b8d3e66964deb7bb  
02cd35a749435c1985a19d40b997e08be23b91209376641084b00  
b60a13ece004c0381dba3e72e20fda6c80f9af1f018fc80934aab  
4b7f91f306b  
←7c818384818028db279f2e0014980bcfa98ab5fb5839c74b72a78d0  
7cc7507bb50c4bd0ad751004220fcd7ed22a9d25bd268289aba98  
dfc5548c207e724d24ceb31eefef060202d98c5be3498c4be8cc9  
21090bfd3e46b110cf4c9282bbf7c2ed9a5d54826e6a1ccdbfa82  
9e0dea1a7a603a797234f7347d9524f9c220c6dae320a41d1076a  
a9000
```

Con la clave pública de la tarjeta y la privada del terminal se calcula el secreto k , a partir del cual se generan las claves de sesión del canal:

```
pukICCDH2->  
28db279f2e0014980bcfa98ab5fb5839c74b72a78d07cc7507bb5  
0c4bd0ad751004220fcd7ed22a9d25bd268289aba98dfc5548c20  
7e724d24ceb31eefef060202d98c5be3498c4be8cc921090bfd3e
```

```
46b110cf4c9282bbf7c2ed9a5d54826e6a1ccdbfa829e0dea1a7a
603a797234f7347d9524f9c220c6dae320a41d1076aa

// calcular k = pukICCDH2^ prkIFDDH2 mod p

?k->
5ba47c4f97b6be2c2e79c01113523cbd1f08d5f7888b13dba3ee9
350f6c23efd265d0db233456ef103e2970c1f0d586dc84ec51994
78991ad011098c60c557a9edc3a6b128859e590c01419ec812349
92f8e00b0d1830922e2e9baa9494781712bf9f29efdd0cc3b429d
642bd949f0399dce982b67c8eb11c0820d0050b0f468
```

5.4.6 Cuarto comando General Authenticate

Se validan las claves de sesión generadas en el paso anterior, por medio de un MAC que calcula el terminal y comprueba la tarjeta, la cual devolverá un segundo MAC.

```
// Calculo de las claves de sesión:

?kenc =sha1(?k.'00000001')
?kenc-> cd18c1d042b510dd198dcd81fe487d07 //16MSB bytes

?kmac = sha1(?k.'00000002')
?kmac-> 0112d0b3a8c3237406f9acbc19f08798 //16MSB bytes
```

Se calcula el Mac del terminal:

```
// ?data = '7f49818f06'. ?oid. '848180'.PukICCDH2;

?data->
7f49818f060a04007f0007020204010284818028db279f2e00149
80bcfa98ab5fb5839c74b72a78d07cc7507bb50c4bd0ad7510042
20fcd7ed22a9d25bd268289aba98dfc5548c207e724d24ceb31ee
fef060202d98c5be3498c4be8cc921090bfd3e46b110cf4c9282b
bf7c2ed9a5d54826e6a1ccdbfa829e0dea1a7a603a797234f7347
d9524f9c220c6dae320a41d1076aa

// cálculo del Mac. AES CMAC-mode
?mac = AES_mac(?data, ?kmac);
?mac-> 0457fad88c6c07dc
```

Se envía el comando General Authenticate y se recupera el MAC devuelto por la tarjeta. El canal PACE está establecido. Se inicializa el contador de secuencia a ceros:

```
?data = '7C0A8508'. ?mac;
?data->0457fad88c6c07dc
```

Establecimiento de canal seguro sin contactos

```
->008600000c7c0a85080457fad88c6c07dc  
<--7c0a86084a8d8840257d922c9000
```

Comprobamos el MAC devuelto por la tarjeta:

```
// ?data = '7f49818F06'. ?oid. '848180'.PukIFDDH2;  
  
?data->  
 7f49818f060a04007f000702020401028481805ba2de272a6a6cd  
d9a199c8111de5b2cecf0c678c77fe0a26cda8c850c8707793f5  
e544e7c0f7bc92807489d7c34816e194afc6466729c89a68441b7  
e8b8d3e66964deb7bb02cd35a749435c1985a19d40b997e08be23  
b91209376641084b00b60a13ece004c0381dba3e72e20fda6c80f  
9af1f018fc80934aab4b7f91f306b  
  
// cálculo del Mac. AES CMAC-mode  
?mac = AES_mac(?data, ?kmac);  
?mac->4a8d8840257d922c
```

Ya tenemos el canal PACE establecido:

```
// CANAL SEGURO ESTABLECIDO  
-->Secure Channel Keys (AES)  
-->kenc=cd18c1d042b510dd198dcd81fe487d07  
-->kmac=89fc6e0275d78a6c8e5a91f9647b0ba7  
-->ssc=0112d0b3a8c3237406f9acbc19f08798  
-->macLen=8
```

A partir de este momento, aunque trabajemos con el interfaz sin contactos, hay que establecer el canal seguro tal y como se indica en el apartado 6 Establecimiento de canal seguro.

6 Establecimiento de canal seguro

Para realizar el establecimiento de canales seguros se han de invocar las siguientes funciones:

1. Get Chip Info,
2. Select File (6020),
3. Get Response,
4. Read Binary (Lectura del certificado de autoridad intermedia para componente)
5. Select File (3F00),
6. Get Response,
7. Select File (601F),
8. Get Response,
9. Read Binary (Lectura del certificado de componente),
10. Manage Security Environment (Selección de la CA Raíz),
11. Perform Security Operation (Verificación del certificado autoverificable de la CA intermedia),
12. Manage Security Environment (Selección de clave en memoria y modo de uso),
13. Perform Security Operation (Verificación del certificado autoverificable del terminal),
14. Manage Security Environment (Selección de claves para autenticación),
15. Internal Authenticate,
16. Get Response,
17. Get Challenge,
18. External Authenticate.

Si trabajamos con el interfaz sin contactos, antes de realizar estos pasos, hay que establecer el canal seguro mediante el protocolo PACE tal y como se indica en el apartado 5 Establecimiento de canal seguro sin contactos.

7 Ejemplo de establecimiento de canal seguro¹ de usuario.

7.1 Datos externos a la tarjeta

Para establecer un canal seguro de usuario con una tarjeta DNIE se necesita utilizar los siguientes datos:

- Clave pública de la autoridad certificadora raíz de la jerarquía de los certificados de componente.

```
Módulo =  
EADEDA455332945039DAA404C8EBC4D3B7F5DC869283CDEA  
2F101E2AB54FB0D0B03D8F030DAF2458028288F54CE552F8  
FA57AB2FB103B112427E11131D1D27E10A5B500EAAE5D940  
301E30EB26C3E9066B257156ED639D70CCC090B863AFBB3B  
FED8C17BE7673034B9823E977ED657252927F9575B9FFF66  
91DB64F80B5E92CD  
Exponente público =  
010001
```

- Certificado verificable por la tarjeta, de la autoridad certificadora intermedia.

```
C CV CA =  
7F2181CE5F37818062BA9D0F4B6C4759DB77A3FDB2CF94D8  
61AF274559E5C7A96A62597E1B325B3401B286BEE75786BB  
B216C5D2870108EBFECD85AD6995390A0B4C31EC2E50725E  
1133FAB18BB582757853E06DAB63DB84040B009ABDC40275  
72BD2CA7882E836FAA8EF22F89CBD005B205700E792CE90B  
1D915FD14EA7F905FBB7D719F2A71E675F383D2DE008ED47  
BDE7094C9CE2DB4C1E1994B22C3635CE299F25A9AF7387AC  
E63F1AF978E13524BCAF1AD27E567D1E7AE018C196D05F38  
972DF6CD0001000142086573524341600005
```

- Identificador de la anterior autoridad certificadora (campo CHR del certificado)

```
CHR =  
000000006573534449600006
```

¹ Si empleamos el interfaz sin contactos, anteriormente a estos pasos hemos debido ejecutar la secuencia descrita en el apartado 5 Establecimiento de canal seguro sin contactos.

Ejemplo de establecimiento de canal seguro de usuario.

- Certificado verificable por la tarjeta, del Terminal que va a establecer el canal de usuario.

```
C CV IFD =
7F2181CD5F37818053FC116560421DF10E31EAC50B4C1ACF
FBD79A2C33E518A074B4BA3E2DCA924988D0C333520B2379
C0F147C2223C8B92C5E035D98D207727E3E6383B8AF36237
CF8F1CAB978CA4A76AC321CEC5564E29E7492321126D9AA2
EC21BDF7B4C96087A6ACA2CA184AF60C90CBA86F94C0D52D
4D18F9D396FA66AFFC16160462E2F5B05F383CE7A8550A80
21C3443DE26904B29CDE69748C764C54E76A047D2F82A526
23A520CC802636C56C9D1BB2E5B8B72B5EC4B44858535B27
9F239B0001000142086573544341600005
```

- Identificador del Terminal, incluido en el campo CHR del certificado.

```
CHR:
000000002000000000000001
```

- Clave privada del Terminal asociada al certificado anterior.

Módulo:

```
Módulo:
cafca9535ef31aff7500a41f5926ad9583e8615c9fae9874
2e32b57b46d582ed4032ea77de64502aa1ccdc81a1b7a831
586b0827bf7264f8b78fec847c5b7f860e64bf5210bef58d
e7a8550a8021c3443de26904b29cde69748c764c54e76a04
7d2f82a52623a520cc802636c56c9d1bb2e5b8b72b5ec4b4
4858535b279f239b
Exponente público:
010001
Exponente privado:
18B44A3D155C61EBF4E3261C8BB157E36F63FE30E9AF2889
2B59E2ADEB18CC8C8BAD284B9165819CA4DEC94AA06B69BC
E81706D1C1B668EB128695E5F7FEDE18A908A3011A646A48
1D3EA71D8A387D474609BD57A882B182E047DE80E04B4221
416BD39DFA1FAC0300641962ADB109E28CAF50061B68C9CA
BD9B00313C0F46ED
```

7.2 Establecimiento de canal seguro de usuario

7.2.1 Reiniciar la tarjeta

En este primer paso, reiniciamos la tarjeta para que la tarjeta pierda cualquier estado previo.

7.2.2 Petición del número de serie de la tarjeta:

Para ello utilizamos el comando 'Read Chip Info':

```
→ 90b8000007  
← 0203cc950536219000
```

El número de serie de la tarjeta es: 0203cc95053621

7.2.3 Selección y lectura de los certificados de componente y de autoridad intermedia de componente:

Empezamos seleccionando el fichero del certificado de la autoridad de certificación intermedia, que reside en el fichero 3F00/6020. Utilizamos los comandos 'Select File' y 'Get Response':

```
→ 00a40000026020  
← 610e  
→ 00c000000e  
← 6f0c850a016020042c00ffff80809000
```

En la respuesta al comando de selección del fichero vemos que el tamaño del mismo es 0x042C bytes, así que lo leemos mediante varios comandos 'Read Binary':

```
→ 00b00000ff  
← 3082042830820391A00302010202101AED357A7BA8C87B43  
FADF3FA9775680300D06092A864886F70D01010505003081  
87310B300906035504061302455331283026060355040A0C  
1F444952454343494F4E2047454E4552414C204445204C41  
20504F4C49434941310D300B060355040B0C04444E494531  
1C301A060355040B0C134143205241495A20434F4D504F4E  
454E5445533121301F06035504030C1830303030303030  
36353733353234343439363030303036301E170D30363032  
32313039333730335A170D3331303232303137313331355A  
3072310B300906035504061302455331283026060355040A  
0C1F444952454343494F4E2047454E9000  
→ 00b000ffff  
← 4552414C204445204C4120504F4C49434941310D300B0603  
55040B0C04444E4945310D300B060355040B0C04464E4D54  
311B301906035504030C12414320434F4D504F4E454E5445  
532030303130819F300D06092A864886F70D0101050003  
818D0030818902818100DE3CC30B668CB2353485FD42A119  
2125F4A82504197EBD3CB6ABFFB8E68F2E84293BB725FD61
```

Ejemplo de establecimiento de canal seguro de usuario.

```
8FD10FDA1A409737D74BD1B97984685156CEFF55F5E3525B
C58BFD3799C2A7E300471D9303105EE8EC4C67D476DC48D1
834670C27EBF4668779D7B855B44AFA733DA451BB954994A
718910FDE6313AAB6266313FDBD1564FA57D0203010001A3
8201A7308201A330120603551D13019000
→ 00b001feff
← 01ff040830060101ff020100301d0603551d0e041604143f
320e9794b8f9566ea3d721a61754fa923a12b1301f060355
1d2304183016801445d76465f22504d8045e6627419d7650
05a2d158300e0603551d0f0101ff04040302010630370603
551d200430302e302c0604551d20003024302206082b0601
05050702011616687474703a2f2f7777772e646e69652e65
732f647063308201020603551d1f0481fa3081f73081f4a0
81f1a081ee8623687474703a2f2f63726c732e646e69652e
65732f63726c732f41524c434f4d2e63726c8681c66c6461
703a2f2f6c6461702e646e69652e65732f434e3d43524c2c
434e3d303030303030303030303030363537339000
→ 00b002fdff
← 353234343439363030303030362c4f553d4143253230524149
5a253230434f4d504f4e454e5445532c4f553d444e49452c
4f3d444952454343494f4e25323047454e4552414c253230
44452532304c41253230504f4c494349412c433d45533f61
7574686f726974795265766f636174696f6e4c6973743f62
6173653f6f626a656374636c6173733d63524c4469737472
69627574696f6e506f696e74300d06092a864886f70d0101
050500038181008f837ad2f2a8c4ba6ab45e0d4b392d6ec0
e66b9da3f97efb08da7dc3bbd3a6549be10471ff7f28209a
17f0717af6f6d51e9f30dfd58fe804f02bf2b25820def595
c2ed85e3b080380438d824702180469000
→ 00b003fc30
← 2667f29d75b30cb9f2bb4a03dda58de2c150ed316ab9b29c
a63210017dd70ffc21316f88f753904d743b70214523760a
9000
```

Realizamos la misma operación de selección y lectura con el certificado de componente ubicado en el fichero 3F00/601F:

```
→ 00a4000002601f
← 610e
→ 00c000000e
← 6f0c850a01601f032500ffff80809000
→ 00b00000ff
← 308203213082028AA00302010202106F58D99A5FF9E72455
2E34E54BABA9E0300D06092A864886F70D01010505003072
310B300906035504061302455331283026060355040A0C1F
444952454343494F4E2047454E4552414C204445204C4120
504F4C49434941310D300B060355040B0C04444E4945310D
```

Ejemplo de establecimiento de canal seguro de usuario.

```
300B060355040B0C04464E4D54311B301906035504030C12
414320434F4D504F4E454E54455320303031301E170D3135
303431353039353233375A170D3236303431353039353233
375A3026310B300906035504061302455331173015060355
0403130E303230334343393530353336323130819F300D06
092A864886F70D010101050003818D9000
→ 00b000ffff
← 0030818902818100B5E170AC5E19E84E20702B3A8171CF00
38908E4DB5E4208D250921ADA62FCA01466380FA6A6D12B0
F26809996CEF7AB61014D8F0A4EAD99DE61AA6C0B0CD32F0
E29F7452509E8535298202FAF1D50F57CE916052D6A1B2A4
AF57C38C5D21D9393BB2A49316B3F2EA0C72ECF545F1D774
7BFB35E87FBE512900F4B364E978A1C70203010001A38201
023081FF300C0603551D130101FF04023000300E0603551D
0F0101FF040403020388301D0603551D0E04160414AABA8F
7841A51B8F6BD2FB39EE95A4392A44F01B301F0603551D23
0418301680143F320E9794B8F9566EA3D721A61754FA923A
12B1303F06082B06010505070101049000
→ 00b001feff
← 333031302F06082B060105050730028623687474703A2F2F
7777772E646E69652E65732F63657274732F41435261697A
2E637274305E0603551D1F045730553053A051A04F864D68
7474703A2F2F63726C732E666E6D742E646E69652E65732F
63726C732F43524C41393732464545463543463546433644
463939393239363833354233313843384443374243314533
2E63726C300D06092A864886F70D010105050003818100B8
31DC492DEE5A28B825918782B8557C755CE4ACE03217B565
03DB6CDE182CC72080D531E8B46195CB7FE9DAA649567864
C341472260A7DAD7406301C7058E79C925891B53B04DA937
DC9DE4998B0B168A4EC06EE7BDC0129000
→ 00b002fd28
← 34BE243D37D771295161A76E0D3EB31735B8ECD708961502
E81374CCB1AB4DF6352DF6CB2C3442659000
```

Una vez leídos ambos certificados se deberá verificar la cadena de certificación de componente utilizando la clave pública de la autoridad certificadora raíz.

Del certificado de componente, extraemos la clave pública de componente que luego tendremos que utilizar para completar las autenticaciones interna y externa.

```
Clave pública de ICC:  
Módulo =  
B5E170AC5E19E84E20702B3A8171CF0038908E4DB5E4208D  
250921ADA62FCA01466380FA6A6D12B0F26809996CEF7AB6  
1014D8F0A4EAD99DE61AA6C0B0CD32F0E29F7452509E8535  
298202FAF1D50F57CE916052D6A1B2A4AF57C38C5D21D939  
3BB2A49316B3F2EA0C72ECF545F1D7747BFB35E87FBE5129
```

```
00F4B364E978A1C7
Exponente público =
010001
```

7.2.4 Selección y carga de la cadena de certificados verificables en la tarjeta

Seleccionamos en la tarjeta la clave pública de la autoridad certificadora raíz de la jerarquía de certificados verificables por la tarjeta. Para ello utilizamos el comando 'Manage Security Environment' utilizando la referencia del fichero donde reside la clave (020f):

```
→ 002281b6048302020f
← 9000
```

Enviamos el certificado verificable por la tarjeta de la autoridad intermedia (**C CV CA**). Para ello utilizamos el comando 'Perform Security Operation':

```
→ 002a00aed27f2181ce5f37818062ba9d0f4b6c4759db77a3
fdb2cf94d861af274559e5c7a96a62597e1b325b3401b286
bee75786bbb216c5d2870108ebfec85ad6995390a0b4c31
ec2e50725e1133fab18bb582757853e06dab63db84040b00
9abdc4027572bd2ca7882e836faa8ef22f89cbd005b20570
0e792ce90b1d915fd14ea7f905fbb7d719f2a71e675f383d
2de008ed47bde7094c9ce2db4c1e1994b22c3635ce299f25
a9af7387ace63f1af978e13524bcacf1ad27e567d1e7ae018
c196d05f38972df6cd0001000142086573524341600005
← 9000
```

Al responder 9000, la tarjeta indica que el certificado ha sido correctamente verificado y que su clave pública ha quedado almacenada en memoria.

A continuación, seleccionamos la clave pública recién cargada con el comando 'Manage Security Environment' y utilizando la referencia del certificado anterior (**CHR**):

```
→ 002281b60a83086573544341600005
← 9000
```

Enviamos el certificado del Terminal con privilegios de canal de Usuario, verificable por la tarjeta (**C CV IFD User**), utilizando el comando 'Perform Security Operation':

```
→ 002A00AED17F2181CD5F37818053FC116560421DF10E31EA
C50B4C1ACFFBD79A2C33E518A074B4BA3E2DCA924988D0C3
33520B2379C0F147C2223C8B92C5E035D98D207727E3E638
3B8AF36237CF8F1CAB978CA4A76AC321CEC5564E29E74923
21126D9AA2EC21BDF7B4C96087A6ACA2CA184AF60C90CBA8
6F94C0D52D4D18F9D396FA66AFFC16160462E2F5B05F383C
E7A8550A8021C3443DE26904B29CDE69748C764C54E76A04
7D2F82A52623A520CC802636C56C9D1BB2E5B8B72B5EC4B4
4858535B279F239B0001000142086573544341600005
← 9000
```

Este certificado de Terminal es el definido en [EN14890-1]. Los certificados con CPI=4 son utilizados para la carga de la clave pública de un terminal, que será utilizada a continuación en un proceso de autenticación. Su formato, una vez recuperado con el uso de la clave de la CA_intermedia, es el siguiente:

| | | | | |
|---------|---------|------------------|---------|---|
| '7F 21' | '81 CD' | C_CV Certificate | | |
| | | '5F 37' | '81 80' | Longitud de la firma (ejemplo para claves de 1024 bits) '6A' = Relleno según ISO 9796-2 CPI = '04' CAR = Defined by Card Manufacturer CHR = Certificate Holder Reference CHA = AID[1..6] Role Identifier OID = '2B 24 07 02 01 04' PK_part1 = 'xx..xx' (MSB..LSB) Hash = 'xx..xx' 'BC' Los campos descritos no están en claro en el certificado, si no que son la entrada de la firma incluida en esta sección. |
| | | '5F 38' | '3C' | PK_part2 = 'XX ... XX' (MSB..LSB) '00 01 00 01' Incluye el resto de la clave pública, que no se pudo poner en PK_part1 (resto del módulo y exponente público en 4 bytes) |
| | | '42' | '08' | CAR Se incluye también en claro para poder identificar la autoridad que firma el certificado |

La codificación de los campos es la siguiente:

CPI es el identificador de la plantilla con que está construido el certificado (Certificate Profile Identifier). Este campo ocupa un solo byte.

CAR es un campo de ocho bytes, que identifica la autoridad certificadora que emitió el certificado. (Certification Authority Reference)

CHR es el identificador del propietario del certificado (Card Holder Reference), en este caso, corresponde al número de serie del terminal propietario del certificado. El campo ocupa siempre doce bytes. Si el número de serie es de menor longitud, se completará con ceros a la izquierda. Se considera que el número de serie es al menos de ocho bytes.

CHA indica los niveles de autorización que se conceden al propietario del certificado (Certificate Holder Authorisation). Es un campo de siete bytes, en los que los seis primeros son la parte más significativa del identificador de aplicación definido en [EN14890-1] (A0 00 00 01 67 45), y el último byte es el identificador de 'rol' del certificado.

Ejemplo de establecimiento de canal seguro de usuario.

Es en este campo de rol del certificado donde se habilita la funcionalidad de canal de Usuario cuyo valor deberá ser 0x02 para tener privilegios de firma de datos.

OID es un campo de seis bytes con el identificador de objeto definido en [EN14890-1]. Identifica la función resumen empleada. Su valor debe ser 2B 24 07 02 01 04 para SHA-256.

PK_part1 es la primera parte de la clave pública incluida en el certificado. La clave pública RSA está formada por el módulo, concatenado con el exponente público codificado en 4 bytes. En este campo se incluirá el máximo número de bytes posible (en función del tamaño de las claves) y el resto deberá ir en claro en el campo PK_part2.

Hash es el resultado de aplicar la función resumen a la concatenación de los campos CPI, CAR, CHR, CHA, OID, y la clave pública completa.

PK_part2 es el campo con la parte de la clave pública que no se pudo incluir en PK_part1.

El bloque de datos '6A' || CPI || CAR || CHR || CHA || OID || PK_part1 || Hash || 'BC' constituye la parte a firmar del certificado, y deberá tener el mismo tamaño que la clave RSA con la que está firmado el certificado.

Seleccionamos la clave pública recién cargada para autenticación. En el mismo comando 'Manage Security Environment' aprovechamos para seleccionar en la tarjeta la clave privada de componente (referencia '021F', bajo el tag '84'), y el identificador del algoritmos (bajo el tag '80', en el ej. 'FF44204C' que corresponde a Autenticación del terminal con SHA-256 y establecimiento de canal con AES-128 bits, CMAC, incluyendo SSC para el cifrado y utilizando SHA-256 para la derivación de claves).

```
→ 0022c1a418830c000000011223344556677888402021f
   8004ff44204c
← 9000
```

7.2.5 Autenticación interna (de la tarjeta)

Enviamos el comando 'Internal Authentication', mediante el que se le pide a la tarjeta que demuestre que posee la clave privada asociada a su certificado de componente. Para este comando el Terminal deberá generar un valor aleatorio de 8 bytes. (En el ejemplo RND.IFD = 'DBC3B533A949906F', y sn.IFD = '1122334455667788'):

```
→ 0088000010dbc3b533a949906f1122334455667788
← 6180
→ 00c0000080
← 0E35B9B256F86E47DD48D5ED55E36FD0A27076D8597C8537
   91EB3B16E664F4FC712D852B8A333B958940F62043693CAF
   A49A1212F46B14DE6FD8BE2816DDD95F729CBFB78E470D4D
   CB6E8A33D8571E6C72CF4FD442DBD419B64C2D97BBC369EE
   FCB3EF1D9991DF9D6806416498608AE7E3D0BE0DF3C368A5
```

Ejemplo de establecimiento de canal seguro de usuario.

```
6D55AD2C4B143B3D9000
```

La respuesta de la tarjeta es un mensaje encriptado con la clave privada de componente de la tarjeta, a continuación se le ha aplicado la función SIGMIN y por último encriptado de nuevo con la clave pública del Terminal:

```
E[PK.IFD.AUT] (SIGMIN)

Donde SIGMIN = min (SIG, N.ICC - SIG)

y

SIG= DS[SK.ICC.AUT]
(
  '6A' = relleno según ISO 9796-2 (DS scheme 1)
  PRND1 = 'XX ... XX' bytes de relleno aleatorios generados en la tarjeta. La
    longitud debe ser la necesaria para que la longitud desde '6A'
    hasta 'BC' coincida con la longitud de la clave RSA
  Kicc = Semilla de 32 bytes, generada por la tarjeta, para la derivación de
    claves del canal seguro.
  h[PRND1 || KICC || RND.IFD || SN.IFD ] = hash SHA-256 que incluye los
    datos aportados por la tarjeta y por el terminal
  'BC' = relleno según ISO 9796-2 (option 1)
)
```

Por lo tanto, para verificarlo hay que empezar por desencriptarlo con la clave privada del Terminal:

```
Mensaje entregados por la tarjeta:
0E35B9B256F86E47DD48D5ED55E36FD0A27076D8597C8537
91EB3B16E664F4FC712D852B8A333B958940F62043693CAF
A49A1212F46B14DE6FD8BE2816DDD95F729CBFB78E470D4D
CB6E8A33D8571E6C72CF4FD442DBD419B64C2D97BBC369EE
FCB3EF1D9991DF9D6806416498608AE7E3D0BE0DF3C368A5
6D55AD2C4B143B3D

Mensaje desencriptado con clave privada de Terminal:
0EBCDB531194A0DDA3A17AEE3D69FB8DE70AEEE9AB38F9E9
613D361CE1ACEAEBFAA7440B36AE78365F5FAC9E42EF0FFB
A348BF550F6BDCF8B5063C0332BF797FD3F61F05BA7D2518
F8CEE1935F1DEB5E8DD6D6ECED44E54B673537755D81567
F9C351B8080A33B134BCB7817F7176F09E917CDE5ACD02A8
C8852F1B773C1AEE
```

Ahora lo que tenemos es el resultado de la función SIGMIN, que puede ser directamente SIG o bien N.ICC-SIG. Probamos primero si es el primer caso y por tanto lo desencriptamos directamente con la clave pública de componente de la tarjeta:

```
Desencriptado con clave pública de tarjeta:
6A39CDA695632C007329477912CAA98D48AA894E6AFFFEA0
F3FA0F069579CF4645E89CFD2CDD72F0D168DFABC249F3AD
20EB529199C2B707194FD66C04293D5AEF062AFD9F0DBAC9
113A9E6D7AB288CC4E455DAA098DB8D18C8237E772678882
78D27F763F3F4AB76495152A84E908A8D66ACFD614FE5E32
```

Ejemplo de establecimiento de canal seguro de usuario.

4C5C85BEA0D803BC

Sobre este mensaje descriptado hay que comprobar si se ajusta al formato esperado y si coincide el “hash”. En este ejemplo sí coincide con el formato esperado (debe empezar por 0x6A, y acabar con 0xBC). En caso de no coincidir con el formato pasamos a probar si el resultado de la función SIGMIN fue N.ICC-SIG. Calcularíamos N.ICC-SIGMIN(..) y descriptaríamos de nuevo con la clave pública de la tarjeta:

N. ICC-SIGMIN (..)
 ..
Descriptado con clave pública de tarjeta:
 6axxxx..xxxxbc

En este caso sí que obtenemos el relleno correcto, así que pasamos a descomponerlo de acuerdo a la estructura del mensaje y comprobamos el “hash”:

| | |
|---|--|
| '6A' = relleno según ISO 9796-2 (DS scheme 1) | 6A |
| PRND1 = 'XX ... XX' bytes de relleno aleatorios generados en la tarjeta. La longitud debe ser la necesaria para que la longitud desde '6A' hasta 'BC' coincida con la longitud de la clave RSA | 39CDA695632C007329477912CAA98D48AA894E6AFFFEA0F3FA0F069579CF4645E89CFD2CDD72F0D168DFABC249F3AD20EB529199C2B707194FD66C04293D |
| Kicc = Semilla de 32 bytes, generada por la tarjeta, para la derivación de claves del canal seguro. | 5AEF062AFD9F0DBAC9113A9E6D7AB288CC4E455DAA098DB8D18C8237E7726788 |
| h[PRND1 KICC RND.IFD SN.IFD] = hash SHA-256 que incluye los datos aportados por la tarjeta y por el terminal | 8278D27F763F3F4AB76495152A84E908A8D66ACFD614FE5E324C5C85BEA0D803 |
| 'BC' = relleno según ISO 9796-2 (option 1) | BC |

Datos sobre los que calcular el hash:
 39CDA695632C007329477912CAA98D48AA894E6AFFFEA0F3FA0F069579CF4645E89CFD2CDD72F0D168DFABC249F3AD20EB529199C2B707194FD66C04293D

 5AEF062AFD9F0DBAC9113A9E6D7AB288CC4E455DAA098DB8D18C8237E7726788

 DBC3B533A949906F

 1122334455667788
Hash calculado:
 8278D27F763F3F4AB76495152A84E908A8D66ACFD614FE5E324C5C85BEA0D803

Al coincidir el “hash” calculado con el recuperado en el mensaje descriptado, la autenticación interna se considera correcta. El valor de Kicc debemos guardarlo, ya que se utilizará más adelante para el cálculo de las claves de canal.

7.2.6 Autenticación externa (del Terminal)

El siguiente objetivo es realizar la autenticación externa, para lo cual hay que comenzar solicitando a la tarjeta un desafío de 8 bytes (RND.ICC):

```
→ 0084000008  
← B7228891B7EF2C8D 9000
```

Ahora hay que construir el campo de datos para el comando 'External authentication' de acuerdo al siguiente formato:

```
E[PK.ICC.AUT](SIGMIN)  
  
Donde  
  
SIGMIN = min (SIG, N.IFD - SIG)  
  
y  
  
SIG= DS[SK.IFD.AUT]  
(  
  '6A' = relleno según ISO 9796-2 (DS scheme 1)  
  PRND2 ='XX ... XX' bytes de relleno aleatorios generados por el terminal.  
    La longitud debe ser la necesaria para que la longitud desde '6A'  
    hasta 'BC' coincida con la longitud de la clave RSA  
  KIFD = Semilla de 32 bytes, generada por el terminal, para la derivación de  
    claves del canal seguro.  
  h[PRND2 || KIFD || RND.ICC || SN.ICC ] = hash SHA-256 que incluye los  
    datos aportados por la tarjeta y por el terminal  
  'BC' = relleno según ISO 9796-2 (option 1)  
)
```

Generamos PRN2 y Kifd como valores aleatorios de la longitud apropiada:

```
PRN2 :  
AEA7D500DC1A4C5C4BC9BF675C9ED3F116AEFA4B33A2EE55  
0BD900045F78FE320457433666D700B7CEAE8A783EA60DBE  
FD1F51B2877B0786A7F5042B14B8  
  
Kifd:  
D8E25F213F58A31F38316AA922C48A93BAA93C8B1F7A1851  
D0EB746059B613FD
```

Calculamos el "hash":

```
Datos sobre los que calcular el hash:  
AEA7D500DC1A4C5C4BC9BF675C9ED3F116AEFA4B33A2EE55  
0BD900045F78FE320457433666D700B7CEAE8A783EA60DBE  
FD1F51B2877B0786A7F5042B14B8  
  
D8E25F213F58A31F38316AA922C48A93BAA93C8B1F7A1851  
D0EB746059B613FD
```

Ejemplo de establecimiento de canal seguro de usuario.

B7228891B7EF2C8D

000203CC95053621

Hash calculado:
 33EE5BD388F8835D97D36086EA301531941F343F82D4827ED
 22BFD6E47C169D6

Construimos el mensaje en claro:

| | |
|---|--|
| '6A' = relleno según ISO 9796-2 (DS scheme 1) | 6A |
| PRND2 = 'XX ... XX' bytes de relleno aleatorios generados por el terminal. La longitud debe ser la necesaria para que la longitud desde '6A' hasta 'BC' coincida con la longitud de la clave RSA | AEA7D500DC1A4C5C4BC9BF675C9ED3F1 16AEFA4B33A2EE550BD900045F78FE32 0457433666D700B7CEAE8A783EA60DBE FD1F51B2877B0786A7F5042B14B8 |
| KIFD = Semilla de 32 bytes, generada por el terminal, para la derivación de claves del canal seguro. | D8E25F213F58A31F38316AA922C48A93 BAA93C8B1F7A1851D0EB746059B613FD |
| h[PRND2 KIFD RND.ICC SN.ICC] = hash SHA-256 que incluye los datos aportados por la tarjeta y por el Terminal | 33EE5BD388F8835D97D36086EA301531 941F343F82D4827ED22BFD6E47C169D6 |
| 'BC' = relleno según ISO 9796-2 (option 1) | BC |

Este mensaje lo encriptaremos en primer lugar con la clave privada del Terminal:

SIG:

0FF19E076A7B2643DE841785BB035DAE
 D8EC7539CED00DA901A9F33D6430F49B
 C245B4C6CFC692DF4A237CBB97F39771
 6B8F75CAA1986C18276D5F3E22A6F0CF
 A92D1A030DF8F03ABEB6378AF298F915
 CE180029A344E5F2CE4368E92CAEC3BF
 593B58AF7B054407C532974980D8C5AC
 DA3BE83A133BA7AE9CA85B259AC27069

Calculamos N.IFD-SIG para poder obtener SIGMIN, que en este ejemplo coincide con SIG:

N. IFD-SIG:

..

SIGMIN:

0FF19E076A7B2643DE841785BB035DAE
 D8EC7539CED00DA901A9F33D6430F49B
 C245B4C6CFC692DF4A237CBB97F39771
 6B8F75CAA1986C18276D5F3E22A6F0CF
 A92D1A030DF8F03ABEB6378AF298F915
 CE180029A344E5F2CE4368E92CAEC3BF

Ejemplo de establecimiento de canal seguro de usuario.

```
593B58AF7B054407C532974980D8C5AC
DA3BE83A133BA7AE9CA85B259AC27069
```

Por último encriptamos el mensaje con la clave pública de componente de la tarjeta, lo que nos da:

```
Datos autenticación externa:
4779882d7f8669cd9e4a3375047542d6874f191a7c3f2f65
58c827519c7d116cbcd1f56e815f9d5c9aa439852450ef76
a8ad4c469824b8c2a5aca89dda6daadacacaf6b7fc78d9df
5d4c0a1a6b576f90c58a20c5732ddcfjee6e1f555095df8
263d8dbe9c758df68af65c361f9b7e6b7cd53f26a070795d
d8a978a98ef49af8
```

Enviamos el comando a la tarjeta, y si todo está bien calculado, deberá aceptarlo sin error (devolverá 9000):

```
→ 00820000804779882d7f8669cd9e4a3375047542d6874f19
1a7c3f2f6558c827519c7d116cbcd1f56e815f9d5c9aa439
852450ef76a8ad4c469824b8c2a5aca89dda6daadacacaf6
b7fc78d9df5d4c0a1a6b576f90c58a20c5732ddcfjee6e1
f555095df8263d8dbe9c758df68af65c361f9b7e6b7cd53f
26a070795dd8a978a98ef49af8
← 9000
```

7.2.7 Cálculo de las claves de canal

El canal ya está establecido, sólo nos queda calcular las claves de canal.

En primer lugar calculamos Kifdicc como el XOR de los valores de Kifd y Kicc obtenidos anteriormente:

```
Kifd:
D8E25F213F58A31F38316AA922C48A93
BAA93C8B1F7A1851D0EB746059B613FD
Kicc:
5AEF062AFD9F0DBAC9113A9E6D7AB288
CC4E455DAA098DB8D18C8237E7726788
Kifdicc:
820d590bc2c7aea5f12050374fbc381b
76e779d6b57395e90167f657bec47475
```

La clave de encriptación Kenc se obtiene como los primeros 16 bytes del “hash” de Kifdicc concatenado con el valor 00000001:

```
Datos para el hash:
820d590bc2c7aea5f12050374fbc381b
```

Ejemplo de establecimiento de canal seguro de usuario.

```
76e779d6b57395e90167f657bec47475
00000001
hash:
f829b682771eabd530b6b63c0c92d538
091db91d7bdb184639cf9d1b6f3be1a4
Kenc:
f829b682771eabd530b6b63c0c92d538
```

De la misma forma, la clave para el cálculo del mac, Kmac se obtiene como los primeros 16 bytes del “hash” de Kifdicc concatenado con el valor 00000002:

```
Datos para el hash:
820d590bc2c7aea5f12050374fbe381b
76e779d6b57395e90167f657bec47475
00000002
hash:
39b9ac893656ae0ef8c10df67517547f
5f0d16568d707e1a635f94b1cf97493f
Kmac:
39b9ac893656ae0ef8c10df67517547f
```

Por último el contador de secuencia SSC se obtiene concatenando el desafío de la tarjeta (RND.ICC) con el desafío del Terminal (RND.IFD):

```
RND . ICC:
b7228891b7ef2c8d
RND . IFD:
dbc3b533a949906f
SSC:
b7228891b7ef2c8ddbc3b533a949906f
```

7.2.8 Construcción de un mensaje encriptado bajo canal AES128 con SSC en cifrado y CMAC

Como ejemplo de construcción de un mensaje encriptado para su envío a la tarjeta una vez establecido el canal seguro, analizaremos un comando de selección por nombre del fichero maestro (‘Master.File’).

El estado inicial del canal para este ejemplo es el siguiente:

```
Kenc:
f1b0d6449cec48864c1efabb4957d64b
Kmac:
1665a3adcb579053cc5d908720ce4dc1
SSC:
3de0c9658f836888bd352dbf46462f5f
```

Ejemplo de establecimiento de canal seguro de usuario.

El mensaje en claro que queremos enviar es

```
Mensaje en claro:  
00a404000b4d61737465722e46696c65  
CLA:  
00  
INS:  
a4  
P1 P2:  
0400  
P3:  
0b  
Datos:  
4d61737465722e46696c65
```

Como tenemos campo de datos, en primer lugar lo completamos con relleno (7816) hasta obtener una longitud múltiplo de 16 bytes. En este caso nos quedará un bloque de 16 bytes:

```
Datos con relleno 7816:  
4d61737465722e46696c658000000000
```

A continuación encriptamos este mensaje con la clave Kenc, utilizando AES con SSC incrementado y encadenando bloques con CBC y con un valor inicial nulo:

```
Datos encriptados con AES con SSC en modo CBC :  
f5124ee2f53962e86e66a6d234827f0f
```

Estos datos los completamos con un byte 0x01 delante y los encapsulamos dentro del TLV de datos ("tag"=0x87):

```
TLV de datos:  
871101f5124ee2f53962e86e66a6d234827f0f
```

Ahora tenemos que calcular el MAC. Los datos de entrada se construyen siguiendo estos pasos:

- Al byte CLA, le añadimos los bits que indican que el comando está securizado (CLA = CLA | 0x0c).
- Tomamos los bytes de la cabecera CLA, INS, P1 y P2 y los completamos con relleno 7816 hasta 16 bytes.
- Añadimos el TLV de datos.
- Volvemos a completar con relleno 7816 hasta múltiplo de 16 bytes.

En nuestro ejemplo:

```
CLA con los bits de mensaje securizado:  
0c  
Cabecera con relleno 7816:  
0ca40400800000000000000000000000
```


Ejemplo de establecimiento de canal seguro de usuario.

La siguiente figura ilustra este proceso:

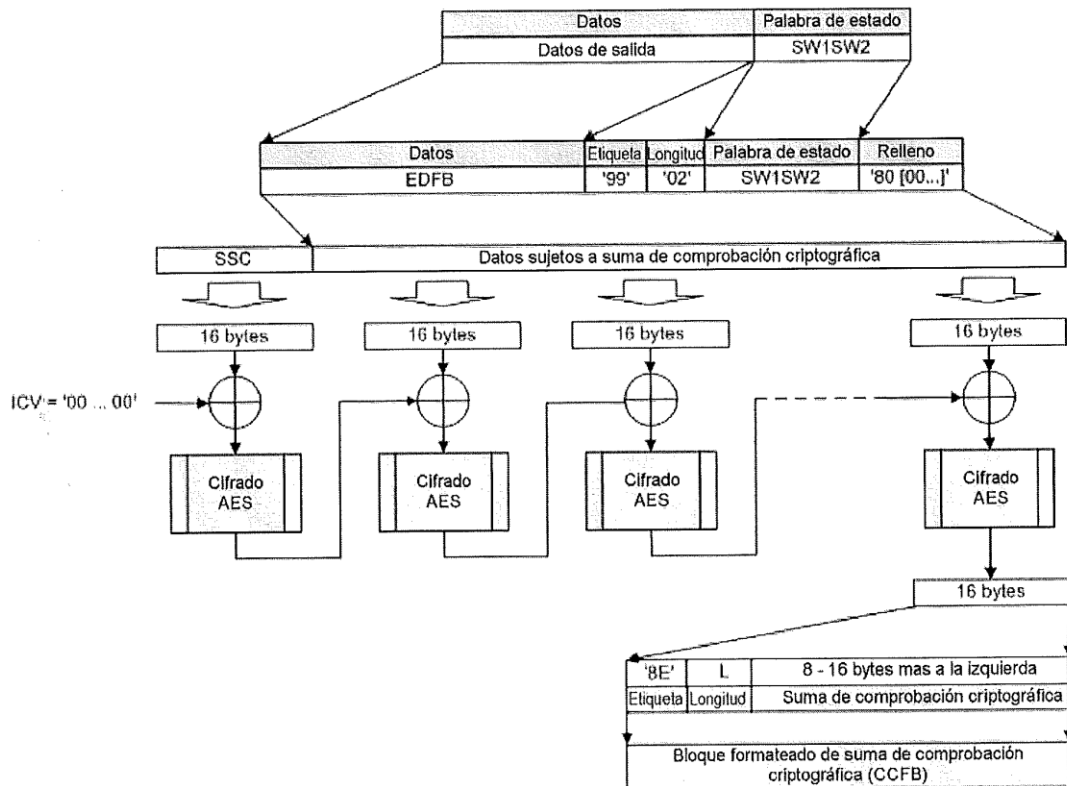


Figura 24 – Ejemplo de cálculo del APDU de respuesta de suma de comprobación criptográfica usando AES con SSC (Modo-CMAC)

Con los datos de nuestro ejemplo:

SSC incrementado:

3de0c9658f836888bd352dbf46462f60

K1:

ab97c270fa027d1193c5e35166c7eb06

CMAC (8 bytes):

70e6de5f679aee64

Construimos el TLV del MAC (“tag” = 0x8e) y el mensaje completo, que incluirá la cabecera (con el CLA indicando mensaje securizado), el TLV de datos y el TLV con el MAC:

TLV del MAC:

8e0870e6de5f679aee64

Mensaje securizado:

```
0ca404001d871101f5124ee2f53962e86e66a6d234827f0f8e087  
0e6de5f679aee64
```

Enviamos este mensaje a la tarjeta:

```
→ 0ca404001d871101f5124ee2f53962e8  
6e66a6d234827f0f8e0870e6de5f679a  
ee64  
← 6131  
→ 00c0000031  
← 87210165c2bf2d95c4234960cbdbaf2e  
fceaa0f1f40808eb4ce8c3b724d898e8  
135c1b990290008e08f9b198242c3286  
099000
```

Para descryptar y comprobar la respuesta de la tarjeta, empezamos descomponiendo los TLV que lo forman:

TLV de datos:

```
87210165c2bf2d95c4234960cbdbaf2e  
fceaa0f1f40808eb4ce8c3b724d898e8  
135c1b
```

TLV con respuesta del comando:

```
99029000
```

TLV con el MAC:

```
8e08f9b198242c328609
```

Para calcular el MAC utilizaremos como entrada todos los TLV devueltos por la tarjeta, excepto el propio TLV del MAC. El procedimiento es el descrito anteriormente:

SSC incrementado:

```
3de0c9658f836888bd352dbf46462f61
```

K1:

```
ab97c270fa027d1193c5e35166c7eb06
```

Datos de entrada del MAC:

```
87210165c2bf2d95c4234960cbdbaf2efceaa0f1f40808eb4ce8c  
3b724d898e8135c1b99029000
```

Con relleno 7861:

```
87210165c2bf2d95c4234960cbdbaf2efceaa0f1f40808eb4ce8c  
3b724d898e8135c1b9902900080000000000000000000
```

MAC calculado:

```
f9b198242c328609
```

Como coincide con el enviado por la tarjeta, consideramos el mensaje correctamente securizado.

Ejemplo de establecimiento de canal seguro de usuario.

Los datos del TLV de datos deberemos descriptarlos y eliminar el relleno 7816 para obtener los datos devueltos por el comando en claro:

```
TLV de datos:  
87210165c2bf2d95c4234960cbdbaf2efceaa0f1f40808eb  
4ce8c3b724d898e8135c1b  
Datos encriptados:  
65c2bf2d95c4234960cbdbaf2efceaa0  
f1f40808eb4ce8c3b724d898e8135c1b  
AES de SSC incrementado:  
477acb7078fe72acdf51e84b8f1787a1  
Descriptando con Kenc:  
6f19840b4d61737465722e46696c6585  
0a383f00000b00111111118000000000  
Eliminando relleno 7816:  
6f19840b4d61737465722e46696c6585  
0a383f00000b0011111111
```

Por tanto, la respuesta equivalente de la tarjeta puesta en claro sería:

```
Respuesta en claro:  
6f19840b4d61737465722e46696c6585  
0a383f00000b0011111111
```

8 Ejemplo de establecimiento de canal seguro² de pin y presentación del pin

La tarjeta DNIE requiere de un canal seguro con derechos para presentación de PIN diferenciado del canal seguro de usuario. El proceso del establecimiento del canal seguro es exactamente igual al descrito en los apartados anteriores, con la diferencia de que el certificado CV que el terminal presenta a la tarjeta tendrá derechos para presentación de PIN.

Enviamos el certificado del Terminal con privilegios de canal de PIN, verificable por la tarjeta (**C CV IFD PIN**), utilizando el comando ‘Perform Security Operation’:

El campo CHA de este **C CV IFD PIN** de acuerdo a EN 14890-1 y **CHA** indica los niveles de autorización que se conceden al propietario del certificado (Certificate Holder Authorisation). Es un campo de siete bytes, en los que los seis primeros son la parte más significativa del identificador de aplicación definido en [EN14890-1] (A0 00 00 01 67 45), y el último byte es el identificador de ‘rol’ del certificado.

Es en este campo de rol del certificado donde se habilita la funcionalidad de canal de PIN cuyo valor deberá ser 0x08 para tener privilegios de canal para presentación de datos.

En este ejemplo asumimos que el canal acaba de ser establecido y las claves del canal son:

```
Kenc:  
598f26e36e11a8ec14b81e19bda223ca  
Kmac:  
5de2939a1ea03a930b88206d8f73e8a7  
SSC:  
d31ac8ec7ba0fe6e
```

El siguiente paso es la presentación del PIN del usuario, ya que es requerido para poder utilizar cualquiera de las claves privadas residentes en la tarjeta. El PIN de la tarjeta utilizada en este ejemplo es ‘CRYPTOKI’, con lo que los comandos de presentación de PIN serán:

```
→ [002000000843525950544f4b49]  
→ 0c20000019871101ce1ab937c332f3faee43336d4311ef33  
8e046908df4e  
← 610a  
→ 00c000000a  
← 990290008e04b4bbf3a69000
```

² Si empleamos el interfaz sin contactos, anteriormente a estos pasos hemos debido ejecutar la secuencia descrita en el apartado 5 Establecimiento de canal seguro sin contactos.

Ejemplo de establecimiento de canal seguro de pin y presentación del pin

←[9000]

NOTA: Los símbolos → y ← indican los comandos antes de encriptar, y los símbolos → y ← indican los comandos securizados que realmente se envían a la tarjeta.

9 Ejemplo de firma con el DNIE 3.0

9.1 Establecimiento del canal de PIN y presentación del PIN

Este apartado se vio con detalle en el apartado 3 anterior de establecimiento del canal de PIN y presentación del PIN.

El requisito de presentación de PIN será necesario para realizar la operación de firma de datos con el certificado de firma.

9.2 Establecimiento del canal de Usuario

Este apartado se vio con detalle en el apartado 2 anterior de establecimiento del canal de Usuario.

9.3 Proceso de firma de datos

Tras la presentación de PIN y el establecimiento del canal de usuario, vamos a describir el proceso y los comandos que intervienen en una operación de firma de datos con la clave de firma.

En este ejemplo asumimos que el canal acaba de ser establecido y las claves del canal son:

```
Kenc :  
7252e29d5ab69ed89924263128e7e997  
Kmac :  
e7fa91b49f6e83703a0f9c63708bcd6d  
SSC :  
44ced6da57b5ae2af1f13263f6a104d9
```

El siguiente paso es seleccionar la clave con la que queremos firmar. Para ello necesitamos conocer su identificador en la tarjeta. La forma de obtenerlo es analizando la estructura PKCS15 de la tarjeta. En el fichero PrKDF (en el path 3f00/5015/6001) encontraremos la información de las claves privadas almacenadas en la tarjeta. Los datos que nos interesan son su identificador de PKCS11 (CKA_ID), su etiqueta de PKCS11 (CKA_LABEL) y el path donde reside la clave. En nuestro ejemplo en el PrKDF tendremos descritas dos claves:

```
Primera Clave:  
CKA_LABEL = KprivAutenticacion  
CKA_ID = A00038351420150415122915  
Path = 3F110101
```

Ejemplo de firma con el DNIE 3.0

```
Segunda Clave:  
CKA_LABEL = KprivFirmaDigital  
CKA_ID = F00038964820150415122927  
Path = 3F110102
```

Para simplificar este ejemplo, no se incluye la secuencia de comandos para seleccionar, leer y analizar el PrKDF. Para ello únicamente es necesario utilizar los comandos de ‘Select File’ y de ‘Read Binary’. Su contenido se ajusta a la estructura ASN.1 descrita en el PKCS15 para este fichero.

La clave que queremos utilizar es la de autenticación, que por la etiqueta vemos es la primera. El identificador de la clave en la tarjeta coincide con el último byte del “path” del fichero donde se encuentra, que en nuestro caso es el 01.

Con todo esto, utilizaremos el comando ‘Manage Security Environment’ para seleccionar la clave 1 en la plantilla de firma:

```
→ [002241b60484020101]  
→  
0c2241b61d8711015ff93d68258d95caa71d229d3816146b8e081  
c745e75f8955e85  
← 610e  
→ 00c000000e  
← 990290008e084aef9fc845cfac999000  
← [9000]
```

Los datos que queremos firmar son:

```
Datos a firmar:  
"Datos a firmar"  
Hash SHA_256 de los datos a firmar:  
99c7ec629a3e77950dbca1a857ea2deecc42cf9750e7872982374  
7493be657fa
```

Por último utilizaremos el comando ‘Perform Security Operation’ para firmar el “hash” de los datos. En el comando, además del “hash”, también incluimos el “DigestInfo” del SHA_256 para que sea incluido en la firma:

(3031300d060960864801650304020105000420):

```
→  
[002a9e9a333031300d06096086480165030402010500042099c7  
ec629a3e77950dbca1a857ea2deecc42cf9750e78729823747493  
be657fa]  
→  
0c2a9e9a4d874101f2737a8586809dc361a615d25c297469ee9a1  
921be876003c76ed6e061a9c15d5945f5ddd801952f590f9e41a9  
251379e1719e3fa8fc9db8e81f2fc1c9f4d42e8e085dd1698197b  
ea4d1  
← 6100
```



```
→ 00c0000000
← 87820111017a6082cd41a425f0d57f6c00d5f31f7b1de7fdbe43a
9eec9113bc396c5dc8963d634df06708dd022839279f1be4bfe05
220c88d1aa14e0123fba7d81a838c86a0f62f1fea259ed5042ca0
faab2f2afa35307cd4be3acf5e5b5c136555bd965aeb6dcdfefe1
4fa6dfab8591ee8371b62ca9bb4325e391a1a719eafb13412810b
287295e4df021c91878ed7841c414be57032c5bad7dfd7b9ec009
b928be00a9e279e26afbe2cf36d2c324297fe0e4497eaa33c141c
df8d27d1c8a0f679483ae095d6bf205ad529abe19c75842148271
5b462e12bc3265a895a912789c8201d41f9cdedb11b34cac65252
6b33463e4762e384fe6d6a5a4d7c02db6cf6123

→ 00c0000023
← 1ba9f7e77f9e1d27664360427241eec3b81c181450990290008e0
8d65f79bf992eaedd9000
← [585a9791828326cd248ca3b35aa2df1e386b1bf4f6a9ff6147cf
318b67b18062df61f8433ddb0d493e6943b0598727722769ae37f
17f56f836017ac6be9369f0ec73178ea4d522bb5466b64ba71f18
0064626b815ffe8b2b973d83dfe08789fd58d8e4906805e1fa027
76158dcc0be69bc5d0a2ae861aabc0f1330be08dc8ae899fb527b
1505db71568449fdbelc335c4aa23a48a43f8ffba3c6a9dbf9066
e92ced77ebdf51b75dd353e58866b1e75a678f984c171bcac83a8
6fe1d2a217f07529591354fc2342018e24173d629e400729de62b
752b8eb940a9cb033a5405ebcc8401e68b437c9ee105432fe4b57
bef14a922d0eec5d6e5646e79383debbf2199000]
```

Por lo tanto, la firma calculada es:

```
Firma del "hash" de los datos:
585a9791828326cd248ca3b35aa2df1e386b1bf4f6a9ff6147cf
318b67b18062df61f8433ddb0d493e6943b0598727722769ae37f
17f56f836017ac6be9369f0ec73178ea4d522bb5466b64ba71f18
0064626b815ffe8b2b973d83dfe08789fd58d8e4906805e1fa027
76158dcc0be69bc5d0a2ae861aabc0f1330be08dc8ae899fb527b
1505db71568449fdbelc335c4aa23a48a43f8ffba3c6a9dbf9066
e92ced77ebdf51b75dd353e58866b1e75a678f984c171bcac83a8
6fe1d2a217f07529591354fc2342018e24173d629e400729de62b
752b8eb940a9cb033a5405ebcc8401e68b437c9ee105432fe4b57
bef14a922d0eec5d6e5646e79383debbf219
```

9.3.1 Lectura del certificado de autenticación y verificación de la firma

Para obtener la lista de certificados disponibles en la tarjeta, en primer lugar será necesario seleccionar, leer y analizar el fichero CDF de la estructura PKCS15 de la tarjeta que está ubicado en la ruta 3F00/5015/6004.

La descripción de la estructura de este fichero puede encontrarse en la especificación del PKCS15.

Los datos que nos interesan son su identificador de PKCS11 (CKA_ID), su etiqueta de PKCS11 (CKA_LABEL) y el “path” donde reside el certificado:

```
Primer certificado:  
CKA_LABEL = CertAutenticacion  
CKA_ID = A00038351420150415122915  
Path = 60617001  
Segundo certificado:  
CKA_LABEL = CertFirmaDigital  
CKA_ID = F00038964820150415122927  
Path = 60617002  
Tercer certificado:  
CKA_LABEL = CertCAIntermediaDGP  
CKA_ID = S00038964820150415122927  
Path = 60617003
```

La relación entre un certificado y su clave privada es que ambos tienen el mismo CKA_ID. En nuestro caso el certificado correspondiente a la clave que hemos utilizado es el primero (CKA_LABEL= CertAutenticacion) que está en la ruta 6061/7001.

Seleccionamos el fichero del certificado (empezando por el fichero maestro) y lo leemos:

```
→ [00a40000023f00]  
→ 0ca400001d871101e89e87b0844efa37b103440a51f81572  
8e08124a18b293e8d628  
← 6131  
→ 00c0000031  
← 872101839fe9295df747790c38116fc7bffffa3d76421726  
b7f44a4f7a216e0a095da4990290008e08a2c5a5ab93a2c7  
129000  
← [6f19840b4d61737465722e46696c65850a383f00000b001  
11111119000]  
  
→ [00a40000026061]  
→ 0ca400001d8711014767281a69cb3bd47706e5fe2c19281  
f8e08d1fffba3697dea89  
← 6131  
→ 00c0000031  
← 8721019197aa51cff5b9a9b3bf549c9b3b5d19da1184bf  
9ff2ebd2ffe68cad479cb104990290008e088bd0621b55  
93716a9000  
← [6f168408444e49652e507562850a3860610008e0ffffff9000]  
  
→ [00a40000027001]
```

Ejemplo de firma con el DNIE 3.0

```
→ 0ca400001d871101b784b982379f257aa9af7b2b417395
  438e082860f684d7645f54
← 6121
→ 00c0000021
← 871101ddaeee9466752b138053ee365c35929a99029000
  8e08ac6e8cb279f0b1769000
← [6f0c850a24700106790020ffffff9000]

→ [00b00000ff]
→ 0cb000000d9701ff8e08d55e3fb50e2418ae
← 61fe
→ 00c00000fe
← 8782010101557af66496d0baf4b9a251499cc55c3c1075a1
  358fa756da46ebf63fb59e75c6812e30557e6cb6bd0c9558
  0e9b769fe076df4d389b744d7005eca79be4860ed58cdb00
  f80bf67229e81431cb6b1e8058da95057656004240f3fc9e
  74dd8ba1e8b19401072a2c582b89db63db131bc4ff99a161
  2e6a83f2eae4f34cbbc9e466c619397793e9552706e29204
  acaaf05a22bb63e49140a00f81d80b5100b5134c96c64908
  81d37719073495292d6b0436bc1d1e88f17c74a2fd641707
  24cdf2fff5fd9da027eaa8fd4a9d2fca5c289db72b481d36
  7b9526ca1a8e9f227253c7dfac43e5d3d087140108ef8d51
  9edeb7119923e58c07adb7b63308f367
← 6113
→ 00c0000013
← 178a67a1ee990290008e08728e52fe373c90469000
← [12060000d1040000789c336862e3336862f9b680998991
  894940a46ccd42c5798d079cdf1cfa5b977ce9b3012f1ba
  7569b47db775e4646565606833c436e034e36e650163661
  26d760430d033510878b47dec533c8d5d9d9d3df4fc1ddd
  5cf35c8d147c1c555c1c75121c0dfc7d3d9d3d190d7801b
  a4929b87c5c5cfd3d550c0800fc2650f080a7575720c361
  43110028930f3703b3a2b80d4281818181ac889f31a9819
  18191a189a999898984601b91606e64616861686c6a6c65
  10665a8ee11321000715885390da120d850d0801f24c6c2
  c3e15a5c907878637e0ecc315a3c2c5ea58979868606fa1
  0ab35602a14600c1d05900a050dc7d01057bf104f9000]

→ [00b000ffff]
→ 0cb000ff0d9701ff8e08795c7c995415969a
← 6100
→ 00c0000000
← 8782010101b5948aed3df079b07fdcdf6109c00d41a66db
  48e4c0785bdf7a3df7a1816f2d21287b29b79c9426334f
  e7cc58a669bdf685b8a03ddb09d462544e7be27bab3561c
  8d7c371abd1ef6ba7a271300dcbe270e1182ea81f119ff6
```

Ejemplo de firma con el DNIE 3.0

```
eb91a5001ec4852a4c96237f95161a21496c1fcb54eac2a
c5ccada5fcc7c83c05389fe0d304c2953d13ec98940863f
cb8ce23e69364f640c89057e3142c83d40bf0e2ac0aedc4
9c592a2e8c50a27c3379e531b68fdea6b5fdf5c95b9a9f0
65ed307c17ac50414c8dcade0c8af408299f416969d3ab7
1eeda8651abdca0c936ea6be40e05c186399cd9a5da8b62
04a75c4c22e699b63c51a5a77e911f857244b4b87b
← 6113
→ 00c0000013
← 658f18391e990290008e08391c46b6d8402f4c9000
← [674767cfc393fd340d9a1895908386919581b989919f01
28cec5d4c4c8c8b06ff6e78f71c7d31e731a5ff86560cd5
1265e98f7aa6de7ac37aec9936766fc95f15aa851e194b3
63e3ebe3bb5bb24fdfe2c979132faee4b25f827fdfd2d32
bb21b99a76e11938c285fa1ab29f446b34771d5fa972d2e
4f8ed9ae484c2cf07d7bb7f479ea7481cbaa2549070fa9c
99d880f3dfffad01665cf2bdcdf166cd2acf4bf20fdfca6
908a445947c617bd5bd35ce6fc397128e0e9e99fafcc34a
55937dff57c9d661da5f38affd309699503ecb27716bdd2
3eb2aceeaade0a0db5944be6f2d5dff7bde2793aed3b8b7
9a2caac9ad899cbb40efef3becc645db8d5dca2dd9000]

→ [00b001feff]
→ 0cb001fe0d9701ff8e086ef934983b361a53
← 6100
→ 00c0000000
← 8782010101793b7f8454d576292f020ad41573ce9634b20
c48f252343639fcabca29bac06251a5eb20fcfe039bca98
ead868175cfbfd970b2172f0b5bbcc86b64400b75f57839
4510a31026b305fde3c29f0327fbe77ff204c1418c9acd8
4d05184553e83bf1feff8aa2fb15ce52938eadf7b5ddf03
a61c2e51f77cf9f326207abfba407a839067fd99ae576cd
225293ef010104cbbbcfd783304d6f124c1eced1828e0c3
d6ea0a218b041fb5ef93479b6b0a3c39e0ddf5c185722bb
5f69fe024ebc3eb4c2f69a178a43cff51b9a968fd4406f1
a4b05155d522eef13f51700c8fd4ccf9be3c498c5c52037
f8acc6cfa4e306fa0bda4250b6cf4de547aee1f46
← 6113
→ 00c0000013
← 188ef1d64a990290008e083ca690e1389038b69000
← [ba578fc1e4fa8cbddae7a55f4ffb6d62c66f8e770f1fea
66cddddd50c631a1dc5bd728504fc6cfb9ebc267ab19ba4
ccc8c0c8c8b9b98e6193431cd32e00106afac3023e37f16
26030670cccbf283782ccc4cec0d06b2203e1f8b188bc8b
273ef7fb2971c487d70bdaea1be58fd55ee12d564037990
b4328b84815883c8974f5f2e6cde39ad52d04b26b4953be
3d0c449e569064a6c1cda6cc0d4c9cec8cc22662062c0c1
```

Ejemplo de firma con el DNle 3.0

```
c6c6c2d0e7c6c80867b11814c0d530b2a4182419a8c3f80
68c6dd21925250556fafaf9c9c5057a0545a5a94989c57a
297999a97aa9c506e670854c6dda5085e5e5e5e8eaf4935
38b4a8af51d9d831233abf4928b4a0c9c41ce56609000]
```

```
→ [00b002fdff]
→ 0cb002fd0d9701ff8e0806ae0b887f1e5c7f
← 6100
→ 00c0000000
← 8782010101e5d8eee384a82544669b89b6a65533113d6ad
ea61dfb8d59db1d49dc5f7abad87aa108001b73729d4d6e468
477915ef3004ae8eabd25e713e99dc913a013ef1fae6d06954
e4c73b71c3d0339c21bbc442227020f7b5af6cb4d8978a104a
b53135fb2923201a21787b685cb4b5dcda58b5aedcf6601de6
dceea6bd6d7bc6e998425b229fc0ab36040b152a9450a1a872
d27af311511f1a7d708df644512794f837d7be71c97fe1ae38
01c014c44f2900a39d520bf58094b085f8517ea39b176d52c6
d71cf8bf1b9458bb49bd41694d3740dedd4d7d6062bb6f65fa
9ff7a95813f87ea995631d305f7c39067145d9bac8968ff3b7
8ad75614c05b965
← 6113
→ 00c0000013
← f34ca3d360990290008e08bbcaa83c3263025e9000

← [b131b032b060e348680d01662f2616031d032d98894c8c
6272784c4c29483668fc007737134be36383c60706464c8c0c
c0e4cb99d0e6c198cacc2c4c8a2106acbc76aff60bd74cc8c
3683f8cb3e6e471f2a3dbb639a3171d2ec5b8f0bf59f6783f4
3092aac70aa81cec68a0720af4b291aa5703146e9c2c8a06f2
c08401f53c27a3a1a004bfa1a5a981a111b030303200822803
2758b0b230b2981998807cc984665b60555adfc3458ffd96fb
fe0bbf5329f06b6d78b8f12455378ed2ffefb2574f4a598256
bc3183b2eea4e6e38f3937edd313719270f896b2694ec9d7eb
caeedfe3cb5520e1fb54e7c125ffe73d2e62d47b88c586cf5
b217dd3b2b31ef9000]

→ [00b003fcdd]
→ 0cb003fc0d9701dd8e08a191f2532baa3387
← 61F2
→ 00c0000000F2
← 8781e101b67724f6c297e7f08418ee37dc408846bf684ab
b9e22b57cc30a212fbc28b4473d6d87774586f33c6b23ebfbd
e8fd861f9d896c19bcb81d1ff698d5e2e99e011addef38d869
78e99fda75ba2bbca50466ae45e839a3cbfb87089cc9bf2f50
98bff8f3391ae33b3c94a75eb3c14e5aebcc0e34e2ade60bba
62a78af97b02f7b48d2ad263ad24b1797e98daa02cd773d359
93b3059d9b1d29c44d32d8b63c0bb9df8a59157e513d6f8f16
```

```
8b5eaccb9965ccac2ad2fa721ca2eeac9deaccbfb5a47103ac
f6495ae52521e5cd47428066edb459641966deaf916b01e6fe
c482b0dbb990290008e082972eafae13dab929000
← [946259ba94fcb2a06ffc371e7a25c5de5a78b27b62e3c2
792faf64a5c9b3b3fd8db0fa6068b3dc67cad534b9fb9d856e
9a87032ea65cecb561fe31ebc2df7b0a96c77d666e0a8f4c8e
1217645dfcf246886623ab24dbdc3995c29172171433ad1cce
be7d56fe994f5567c5e777b13e2e55bea11bba5eb80726ece1
b47ffff48971e7514de3034dbb580b0e3099789dd92ccd259d
9395d9f5654f92dfa9aefdb5f17167eb3e35a57c0d61ed3868
16243f655ed789ea7ea6499ca6158a7e89f35df8428bd3262a
7fe0d35d14e7bed8d1447e6398c9c5863c0d00959001f39000]
```

El contenido del fichero que acabamos de leer es el certificado comprimido con la librería “zlib” y con una pequeña cabecera que indica el tamaño de los datos descomprimidos y el de los comprimidos:

```
Tamaño de los datos descomprimidos:
12060000
Tamaño de los datos comprimidos:
d1040000
Datos comprimidos:
789c336862e3336862f9b680998991894940a46ccd42c5798d0
79cdf1cfa5b977ce9b3012f1ba7569b47db775e464656560683
3c436e034e36e65016366126d760430d033510878b47dec533c
8d5d9d9d3df4fc1ddd5cf35c8d147c1c555c1c75121c0dfc7d3
d9d3d190d7801ba4929b87c5c5cfd3d550c0800fc2650f080a7
575720c36143110028930f3703b3a2b80d4281818181ac889f3
1a981918191a189a999898984601b91606e64616861686c6a6c
6510665a8ee11321000715885390da120d850d0801f24c6c2c3
e15a5c907878637e0ecc315a3c2c5ea58979868606fa10ab356
02a14600c1d05900a050dc7d01057bf104f674767cfc393fd34
0d9a1895908386919581b989919f0128cec5d4c4c8c8b06ff6e
78f71c7d31e731a5ff86560cd51265e98f7aa6de7ac37aec993
6766fc95f15aa851e194b363e3ebe3bb5bb24fdfe2c979132fa
ee4b25f827fdfd2d32bb21b99a76e11938c285fa1ab29f446b3
4771d5fa972d2e4f8ed9ae484c2cf07d7bb7f479ea7481cbaa2
549070fa9c99d880f3dfffad01665cf2bdcdf166cd2acf4bf20
fdfca6908a445947c617bd5bd35ce6fc397128e0e9e99fafcc3
4a55937dff57c9d661da5f38affd309699503ecb27716bdd23e
b2aceeaade0a0db5944be6f2d5dff7bde2793aed3b8b79a2caa
c9ad899cbb40efef3becc645db8d5dca2ddba578fc1e4fa8cbd
dae7a55f4ffbbd62c66f8e770f1fea66cdddd50c631a1dc5bd
728504fc6cfb9ebc267ab19ba4ccc8c0c8c8b9b98e6193431cd
32e00106afac3023e37f1626030670cccbf283782ccc4cec0d0
6b2203e1f8b188bc8b273ef7fb2971c487d70bdaea1be58fd55
ee12d564037990b4328b84815883c8974f5f2e6cde39ad52d04
```

Ejemplo de firma con el DNIE 3.0

```
b26b4953be3d0c449e569064a6c1cda6cc0d4c9cec8cc226620
62c0c1c6c6c2d0e7c6c80867b11814c0d530b2a4182419a8c3f
8068c6dd21925250556fafaf9c9c5057a0545a5a94989c57a29
7999a97aa9c506e670854c6dda5085e5e5e5e8eaf493538b4a8
af51d9d831233abf4928b4a0c9c41ce5660b131b032b060e348
680d01662f2616031d032d98894c8c6272784c4c29483668fc0
07737134be36383c60706464c8c0cc0e4cb99d0e6c198cacc2
c4c8a2106acbc76aff60bd74cc8c3683f8cb3e6e471f2a3dbb6
39a3171d2ec5b8f0bf59f6783f43092aac70aa81cec68a0720a
f4b291aa5703146e9c2c8a06f2c08401f53c27a3a1a004bfa1a
5a981a111b0303032008228032758b0b230b2981998807cc984
665b60555adfc3458ffd96fbfe0bbf5329f06b6d78b8f124553
78ed2ffefb2574f4a598256bc3183b2eea4e6e38f3937edd313
719270f896b2694ec9d7ebcaeedfe3cb5520e1fb54e7c125ff
e73d2e62d47b88c586cf5b217dd3b2b31ef946259ba94fcb2a0
6ffc371e7a25c5de5a78b27b62e3c2792faf64a5c9b3b3fd8db
0fa6068b3dc67cad534b9fb9d856e9a87032ea65cecb561fe31
ebc2df7b0a96c77d666e0a8f4c8e1217645dfcf246886623ab2
4dbdc3995c29172171433ad1cceb7d56fe994f5567c5e777b1
3e2e55bea11bba5eb80726ece1b47ffff48971e7514de3034db
b580b0e3099789dd92ccd259d9395d9f5654f92dfa9aefdb5f1
7167eb3e35a57c0d61ed386816243f655ed789ea7ea6499ca61
58a7e89f35df8428bd3262a7fe0d35d14e7bed8d1447e6398c9
c5863c0d00959001f3
```

Si los datos comprimidos los expandimos con la librería “zlib”, obtendremos el certificado X509 de autenticación del ciudadano:

```
Certificado X509:
3082060e308204f6a00302010202101476aca1219e81c043ecc2
fd7e63d2f3300d06092a864886f70d0101050500306e310b3009
06035504061302455331283026060355040a0c1f444952454343
494f4e2047454e4552414c204445204c4120504f4c4943494131
0d300b060355040b0c04444e49453110300e060355040b0c0750
5255454241533114301206035504030c0b414320444e49452030
3031301e170d3036303231303136343434355a170d3038303732
383138313335335a3076310b3009060355040613024553311230
10060355040513093131313131313131533111300f0603550404
0c0845737061c3b16f6c310d300b060355042a0c044a75616e31
31302f06035504030c2845737061c3b16f6c2045737061c3b16f
6c2c204a75616e2028415554454e544943414349c3934e293082
0122300d06092a864886f70d010105000382010f003082010a
0282010100be9bf3f15ec766e30933d0fa303b087617716eea86
b99aec4563939968fd1c4aa12878426cb8b1ebc7bb846cbcbda0c
6cec5f172244bf180fba5cba86b810395b416195877a82d2912
ec298c21aaafe98444e4c63da86161704deddd75e7659710d325
```

Ejemplo de firma con el DNle 3.0

```
7462c1c2261ec85f55cfebc2b42349d40bf6a0b229794fd01be7
d9122418768868f42eda96449cfcc8c250e5cbf9ea36291b05b3
dd49eb663b5a2cea0ff2c81b24c0071ddca2ea2bc4a67ed52ea8
282664d2371f7bf7beea0ce596f7043761249a7c5d99a62ac1fe
4bd3023b71b53738873b8d2e0034d798bd2bcf1beb96fba898fb
08eee1e12d6a9dbb80760890774b2d32512e1c4e438ad0f33a98
2d0203010001a382029e3082029a300c0603551d130101ff0402
3000300e0603551d0f0101ff040403020780301d0603551d0e04
160414a6ceeff90774c065e0d77e807f7327ea6da42563301f06
03551d23041830168014f4f2f4d0b3b99679114a1c55850b68c2
91927766302206082b0601050507010304163014300806060400
8e4601013008060604008e460104307006082b06010505070101
04643062302706082b06010505073001861b687474703a2f2f6f
6373702e707275656261732e646e69652e6573303706082b0601
0505073002862b687474703a2f2f7777772e707275656261732e
646e69652e65732f63657274732f41435261697a2e6372743043
0603551d20043c303a303806086085540102020204302c302a06
082b06010505070201161e687474703a2f2f7777772e70727565
6261732e646e69652e65732f6470633081f006082b0601050507
01020481e33081e03032020100300b0609608648016503040201
0420553d0e053fe0af1b5c9886305fd34c46c5e122e6dc356891
929bdae3712fe76b3032020101300b0609608648016503040201
0420553d0e053fe0af1b5c9886305fd34c46c5e122e6dc356891
929bdae3712fe76b303a0609608554010202040201300b060960
86480165030402010420553d0e053fe0af1b5c9886305fd34c46
c5e122e6dc356891929bdae3712fe76b303a0609608554010202
040206300b06096086480165030402010420553d0e053fe0af1b
5c9886305fd34c46c5e122e6dc356891929bdae3712fe76b3028
0603551d090421301f301d06082b060105050709013111180f31
393530313230323132303030305a304206086085540102020401
043630343032020102300b06096086480165030402010420517a
668ee1a2e34ea74dfe57dc7910faad5757339225460875ffee6b
ab9264a4300d06092a864886f70d010105050003820101009283
c7e309b2be2e14421840f664b29c74f5d72347874f0c7d1a40f1
2ac8f217a7f992b3b4c40a32043d2e6ba2dec189eca2176671a
1fa652f60fd8e14a625ddaa1c98b9181a19ee9d46a661f0706fd
583af0313ca74c94d5661edf89714629c350d164d18d3c03f89a
d0fdde2039c74c99b25759635a171105a3e9d85429810519069d
9c7913591ed021693a40cdede677f30e252ca8f3ee5d4c447a4d
55b08ae8475160bc093fefe5e43389c52933c082ba0570c00234
4accb31b0a1b6c6a698af4bc624eca8abf7d5f5ecd7ef28264f5
540588c136521f949e8ac87b8f0292093578214e619f440e5573
669123f00e2da25e47a341341fb15634d1806e28
```

Este certificado deberá ser verificado con la cadena de validación de certificados, comprobar caducidad, lista de revocados, etc.

De él podemos obtener la clave pública que nos servirá para comprobar la firma realizada anteriormente:

```
Clave pública de autenticación:  
Módulo =  
be9bf3f15ec766e30933d0fa303b087617716eea86b99aec45639  
39968fd1c4aa12878426cb8b1ebc7bb846bcbda0c6cec5f172244  
bf180fbea5cba86b810395b416195877a82d2912ec298c21aaafe  
98444e4c63da86161704deddd75e7659710d3257462c1c2261ec8  
5f55cfefbc2b42349d40bf6a0b229794fd01be7d9122418768868f  
42eda96449cfcc8c250e5cbf9ea36291b05b3dd49eb663b5a2cea  
0ff2c81b24c0071ddca2ea2bc4a67ed52ea8282664d2371f7bf7b  
eea0ce596f7043761249a7c5d99a62ac1fe4bd3023b71b5373887  
3b8d2e0034d798bd2bcf1beb96fba898fb08eee1e12d6a9dbb807  
60890774b2d32512e1c4e438ad0f33a982d  
Exponente público:  
010001
```

Descriptando con esta clave pública la firma generada anteriormente:

```
Datos firmados con clave de autenticación:  
585a9791828326cd248ca3b35aa2df1e386b1bf4f6a9ff6147cf  
318b67b18062df61f8433ddb0d493e6943b0598727722769ae37  
f17f56f836017ac6be9369f0ec73178ea4d522bb5466b64ba71f  
180064626b815ffe8b2b973d83dfe08789fd58d8e4906805e1fa  
02776158dcc0be69bc5d0a2ae861aabc0f1330be08dc8ae899fb  
527b1505db71568449fdbc1c335c4aa23a48a43f8ffba3c6a9db  
f9066e92ced77ebdf51b75dd353e58866b1e75a678f984c171bc  
ac83a86fe1d2a217f07529591354fc2342018e24173d629e4007  
29de62b752b8eb940a9cb033a5405ebcc8401e68b437c9ee1054  
32fe4b57bef14a922d0eec5d6e5646e79383debbf219  
Firma descriptada:  
0001ffffffffffffffffffffffffffffffffffffffffffffffffffff  
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff  
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff  
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff  
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff  
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff  
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff  
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff  
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff  
ffffffffffffffffffffffffffffffffffffffffffffffffffff003031300d060960  
86480165030402010500042099c7ec629a3e77950dbca1a857ea2  
deecc42cf9750e78729823747493be657fa
```

Donde podemos comprobar que está el “hash” que solicitamos firmar y al que se le ha incluido el relleno PKCS1 hasta completar el tamaño de la clave.

10 Acrónimos

| | |
|----------|--|
| AC | Autoridad de Certificación |
| APDU | Application Protocol Data Unit |
| CAN | Card Access Number |
| CAR | Certification Authority Reference |
| CBC | Cipher Block Chaining |
| CDF | Certificate Directory File |
| CHA | Certificate Holder Authorisation |
| CHR | Certificate Holder Reference |
| CHV | Card Holder Verification |
| CPI | Certificate Profile Identifier |
| CRT | Chinese Remainder Theorem |
| DES | Data Encryption standard |
| DF | Dedicated File |
| DGP | Dirección General de la Policía |
| DIT | Departamento de Documentos de Identificación/Tarjetas |
| DNIe | Documento Nacional de Identidad electrónico |
| EAC | Extended Access Control |
| EEPROM | Electrically Erasable Programmable Read-Only Memory |
| EF | Elementary File |
| FCI | File Control Information |
| FNMT-RCM | Fábrica Nacional de Moneda y Timbre – Real Casa de la Moneda |
| ICAO | International Civil Aviation Organization |
| ICC | Integrated Circuit Card |
| MAC | Message Authentication Code |

| | |
|-------|---|
| MRTD | Machine Readable Travel Document |
| MSE | Manage Security Environment |
| OID | Object IDentifier |
| PACE | Password Authenticated Connection Establishment |
| PAD | Punto de Actualización del DNIE |
| PC/SC | Personal Computer / Smart Card |
| PIN | Personal Identification Number |
| PKCS | Public Key Cryptography Standards |
| RSA | Rivest, Shamir and Adleman |
| SAC | Supplemental Access Control |
| SHA | Secure Hash Algorithm |
| TLV | Tag Length Value |
| XOR | Exclusive OR |

11 Bibliografía

- [BSI-03110] BSI Advanced Security Mechanism for MRTD – EAC. v2.10. March 2012
- [CEN-15480] Identification card systems. European Citizen Card. 2014.
- [DIR] Directiva 1999/93/CE del Parlamento Europeo y del Consejo de 13 de diciembre de 1999 por la que se establece un marco comunitario para la firma electrónica.
- [EN14890-1] Interfaz de aplicación para tarjetas inteligentes utilizadas como dispositivos seguros de creación de firma. Parte 1: Servicios básicos. 2008.
- [EN14890-2] Interfaz de aplicación para tarjetas inteligentes utilizadas como dispositivos seguros de creación de firma. Parte 2: Servicios adicionales. 2008.
- [ISO7816-4] Identification cards -- Integrated circuit cards -- Part 4: Organization, security and commands for interchange. 2005.
- [ISO9796] Information technology -- Security techniques -- Digital signature schemes giving message recovery -- Part 2: Integer factorization based mechanisms. 2002.
- [Ley59/03] Ley 59/2003, de 19 de diciembre, de firma electrónica.
- [PC/SC] PC/SC "Interoperability Specification for ICCs and Personal Computer System" versión 1.0 Diciembre 1997.
- [PKCS#15] Cryptographic Token Information Format Standard. Version 1.1.
- [PKCS#11] PKCS #11 Base Functionality v2.30: Cryptoki
- [PP] Protection profiles for secure signature creation device — Part 2: Device with key generation. Version: 2.0.1. January 2012.
- [SAC] Supplemental Access Control for Machine Readable Travel Documents. Version 1.1. 15 April 2014
- [SHS] Federal Information Processing Standards Publication 180-4 Secure Hash Standard, U.S. Department of Commerce/National Institute of Standards and Technology, March 2012.
- [STDNIE] Declaración de Seguridad de la Tarjeta DNIE 3.0. v1.0 r3. 03/02/2015.