ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE INGENIERÍA

ESTUDIO DE LAS TARJETAS INTELIGENTES (SMART CARDS), ANÁLISIS DE SUS POSIBLES APLICACIONES E IMPLEMENTACIÓN DE UN EJEMPLO DE APLICACIÓN.

PROYECTO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

ROBERTO LUIS CHIMBORAZO TORO

DIRECTOR: ING. FERNANDO FLORES

Quito, Diciembre 2004

DECLARACIÓN

Yo, Roberto Luis Chimborazo Toro, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo el derecho de propiedad intelectual correspondiente a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su reglamento y por la normativa institucional vigente.

Roberto Luis Chimborazo Toro

CERTIFICACIÓN

Certifico que el siguiente trabajo fue realizado por Roberto Luis Chimborazo Toro, bajo mi supervisión.

Ing. Fernando Flores
DIRECTOR DEL PROYECTO

AGRADECIMIENTO

Aprovecho estas líneas para darles las gracias a mi padre Luis Chimborazo quien con su sacrificio diario me ha permitido llegar hasta estas instancias de mi vida y ser quien soy y a mi madre Rosa Toro quien en el poco tiempo que estuvo a mi lado supo ser muy cariñosa y la mejor madre del mundo.

A mis hermanos Ruth, Danny, Dorys y Germán por estar allí cuando los necesité.

A mis mejores amigos que están en mi familia, mis primos.

Al Ing. Fernando Flores por su guía, consejo y paciencia antes y durante la realización de este proyecto.

Roberto Luis Chimborazo Toro

DEDICATORIA

El fruto de este trabajo se lo dedico a quien cree que todo lo bueno que está por venir depende de uno y de las ganas de trabajar por una vida mejor, a todo aquel que sabe que con sus manos puede construir lo que desee construir, a quien mira lo bueno de la vida y lo hace ver aun mejor, el que recoge las malas experiencias y aprende de ellas, quien cree que nada es imposible, en pocas palabras se lo dedico a mi padre.

Roberto Luis Chimborazo Toro

PRESENTACIÓN

Hoy en día las Tarjetas Inteligentes son un elemento muy utilizado por los ciudadanos de otros países, esto por la facilidad de uso de estos elementos y por el gran número de aplicaciones que tienen. Dadas estas características es importante conocer un poco más de esta tecnología que en poco tiempo será muy utilizada por nosotros.

En el presente proyecto se realiza un análisis de los mecanismos utilizados para guardar, extraer y procesar la información de las tarjetas inteligentes. Para ello se analiza, en primer lugar, cuál es el origen de las tarjetas de plástico en general y de las tarjetas inteligentes en concreto, realizándose una descripción breve de sus características físicas y funcionales básicas.

A continuación se realizará una breve descripción de los diferentes tipos de tarjetas inteligentes que existen en el mercado actualmente, de sus características relevantes y de los estándares internacionales.

Los mecanismos de seguridad que se aplican a esta tecnología también son tomados en cuenta en este documento, describiendo los ataques mas frecuentes que pueden afectar a las tarjetas inteligentes y las medidas que se toman para prevenirlos.

Finalmente se analizarán las diferentes aplicaciones de las tarjetas inteligentes en telecomunicaciones, y se realizará el diseño y la implementación de un ejemplo de aplicación

RESUMEN

El proyecto de titulación desarrollado consta de seis capítulos, cuyo contenido se describe a continuación:

En el capítulo I se realiza una corta descripción del desarrollo histórico de las tarjetas inteligentes, estándares internacionales referentes y áreas de aplicación, se estudian también los diferentes tipos de tarjetas inteligentes existentes, así como las diferentes maneras de clasificarlas.

El capítulo II trata a cerca de las propiedades físicas de las tarjetas inteligentes, los formatos de las mismas y los materiales de construcción. Al hablar de propiedades eléctricas se habla de conexiones, fuentes de voltaje, fuentes de corriente, señales de reloj, transferencia de datos, secuencias de activación y desactivación y tipos de memoria con la que cuentan.

En el capítulo III se cubre el desarrollo de los sistemas operativos, diseño y principios de implementación, organización de la memoria y se describe por otro lado el proceso de transferencia de datos desde las tarjetas inteligentes, así como el conjunto de comandos con que cuentan las mismas, los protocolos que se usan, los canales lógicos y procedimientos para una transferencia segura.

En el capítulo IV se describen los diferentes procesos de criptografía usados en las tarjetas inteligentes. Este capítulo es un compendio de los principios esenciales, procedimientos y estrategias que se usan para proteger las tarjetas inteligentes y las aplicaciones de las mismas contra posibles ataques, aquí se habla de mecanismos de ataque y defensa durante el desarrollo, producción y uso de las tarjetas inteligentes.

En el capítulo V se hace una descripción de las aplicaciones que estos elementos tienen en la actualidad y de las posibles aplicaciones que podrían tener en un futuro cercano.

En el capítulo VI se realiza el diseño de un ejemplo de aplicación, para ello se ha elegido una aplicación que permite ilustrar los conceptos previamente estudiados, la aplicación básicamente consiste en almacenar en una memory card el curriculum académico estudiantil.

En la sección de conclusiones y recomendaciones se habla de la aplicación implementada y de las características con las que cuenta, además se realizan recomendaciones de cómo mejorarla y hacerla mas funcional.

Los anexos muestran las líneas de código correspondientes a los programas escritos, características de la tarjeta utilizada en la implementación de la aplicación, características de la biblioteca utilizada para la comunicación del lector de la tarjeta con el computador, y características del lector/programador.

Contenido

Tarjetas Inteligentes

Capítulo 1

1.1 INTRODUCCIÓN 1

1.2 ESTÁNDARES 4

1.2.1 ISO 7810 TARJETAS DE IDENTIFICACIÓN - CARACTERÍSTICAS FÍSICAS (1985) 4

1.2.2 ISO 7811 TARJETAS DE IDENTIFICACIÓN - TÉCNICAS DE GRABACIÓN (1985) 4

1.2.3 ISO 7812 TARJETAS DE IDENTIFICACIÓN- SISTEMA DE NUMERACIÓN Y PROCEDIMIENTO DE REGISTRO PARA IDENTIFICACIÓN DE USUARIOS (1987) 5

1.2.4 ISO 7813 TARJETAS DE IDENTIFICACIÓN - TARJETAS DE TRANSACCIÓN FINANCIERA (1987) 5

1.2.5 ISO 7816 TARJETA INTELIGENTE 6

1.2.5.1 Descripción 6

1.2.5.2 Parte 1: Características físicas de las tarjetas con circuito integrado 6

1.2.5.3 Parte 2: Dimensiones y ubicación de los contactos 7

1.2.5.4 Parte 3: Señales electrónicas y protocolos de transmisión. 8

1.2.5.5 Parte 4: Comandos para el intercambio inter-industrial. 9

1.2.5.6 Parte 5: Sistema de numeración y procedimiento de registro de identificadores de aplicación. 9

1.2.5.7 Parte 6: Elementos de datos 10

1.2.6 EL ESTÁNDAR EMV 11

1.3 CLASIFICACIÓN 12

1.3.1 CLASIFICACIÓN BASADA EN LOS COMPONENTES 13

1.3.1.1 Tarjetas de Memoria o Memory Cards 13

1.3.1.2 Tarjeta con chip 14

1.3.2 CLASIFICACIÓN BASADA EN EL INTERFAZ 16

1.3.2.1 Tarjetas de Contacto 16

1.3.2.2 Tarjetas sin Contacto 17

1.3.2.3 Tarjetas Combi y Tarjetas Híbridas 18

1.3.3 CLASIFICACIÓN BASADA EN LOS SISTEMAS OPERATIVOS 18

1.4 APLICACIONES 19

1.4.1 LA TARJETA INTELIGENTE EN EL ENTORNO DE LAS ENTIDADES FINANCIERAS 19

1.4.2 LA TARJETA INTELIGENTE EN OTROS ENTORNOS 23

1.4.3 ¿PORQUÉ CONSIDERAR LAS TARJETAS INTELIGENTES? 26

1.4.4 VENTAJAS DE LAS TARJETAS INTELIGENTES 26

1.4.5 BARRERAS PARA LA ACEPTACIÓN DE LAS TARJETAS INTELIGENTES 27

Capítulo 2

Características físicas y propiedades eléctricas

2.1 CARACTERÍSTICAS FÍSICAS 29

2.1.1 FORMATOS DE LA TI 30

2.1.2 CARACTERÍSTICAS FÍSICAS ANOTADAS EN ISO/IEC 7816-1 34

2.1.3 LOCALIZACIÓN DE LOS CONTACTOS 37

2.1.4 PROCESO DE FABRICACIÓN DE TIs 39

2.1.5 MATERIALES DE LA TARJETA 44

2.2 PROPIEDADES ELÉCTRICAS 46

2.2.1 CONEXIONES ELÉCTRICAS 48

2.2.2 FUENTE DE VOLTAJE 51

2.2.3 SUMINISTRO DE CORRIENTE 54

2.2.4 RELOJ EXTERNO 57

2.2.5 DESCRIPCIÓN DE LOS CONTACTOS 58

2.2.5.1 Contacto E/S 59

2.2.5.2 Contacto Vpp 59

2.2.5.3 Contacto CLK 60

2.2.5.4 Contacto RST 61

2.2.5.5 Contacto VCC 61

2.2.6 PROCEDIMIENTO PARA LA TRANSMISIÓN DE DATOS 62

2.2.6.1 Secuencias de activación y desactivación 62

2.2.6.1.1 Conexión y activación de los contactos 63

2.2.6.1.2 Reset de la tarjeta 64

2.2.6.1.3 Desactivación de los contactos 67

2.3 MICROCONTROLADORES DE TIs 67

2.3.1 TIPOS DE PROCESADORES 68

2.3.2 TIPOS DE MEMORIAS 71
2.3.2.1 Memoria ROM (read-only memory) 73
2.3.2.2 Memoria EEPROM (electrically erasable programmable read-only memory) 74
2.3.2.3 Memoria Flash EEPROM (flash electrically erasable programmable read-only memory) 75
2.3.2.4 Memoria RAM (random-access memory) 76
2.3.2.5 Memoria FRAM (Ferroelectric random-access memory) 76

Capítulo 3

Sistemas operativos, protocolos de transmisión y comandos de las tarjetas inteligentes

3.1 SISTEMAS OPERATIVOS 78
3.1.1 EVOLUCIÓN HISTÓRICA DE SISTEMAS OPERATIVOS 79
3.1.2 PRINCIPIOS DE DISEÑO E IMPLEMENTACIÓN 80
3.1.3 ORGANIZACIÓN DE LA MEMORIA 83
3.1.4 ARCHIVOS DE LA TI 87
3.1.4.1 Estructura interna de los archivos 89
3.1.4.2 Tipos de archivos 90
3.1.4.2.1 Archivos MF 91
3.1.4.2.2 Archivos DF 91
3.1.4.2.3 Archivos EF 92
3.1.4.2.3.1 Archivos EF de trabajo 92
3.1.4.2.3.2 Archivos EF internos 92
3.1.4.2.4 Archivos de aplicación 93
3.2 TRANSMISIÓN DE DATOS EN TIS 94
3.2.1 ANSWER TO RESET (ATR) 96
3.2.2 PARÁMETRO DE SELECCIÓN DE PROTOCOLO (PPS) 98
3.2.3 PROTOCOLOS DE TRANSMISIÓN DE DATOS 99
3.2.4 TRANSMISIÓN DE DATOS SINCRÓNICA 101
3.2.5 PROTOCOLO DE TRANSMISIÓN T=0 104
3.2.6 PROTOCOLO DE TRANSMISIÓN T=1 108
3.2.6.1 Estructura del bloque 109
3.2.6.2 Contador de secuencia de TX/RX 111
3.2.6.3 Tiempos de espera 111

3.2.6.4 Caracter de tiempo de espera (CWT) 111
3.2.6.5 Bloque de tiempo de espera (BWT) 112
3.2.6.6 Bloque de tiempo de seguridad (BGT) 112
3.2.6.7 Mecanismo del protocolo de transmisión 113
3.2.6.8 Encadenamiento de bloques 113
3.2.6.9 Manejo de errores 114
3.2.7 ESTRUCTURA DE MENSAJE: APDUs 115
3.2.7.1 Estructura de una APDU 116
3.2.7.2 Estructura de una APDU respuesta 118
3.2.8 TRANSMISIÓN DE DATOS SEGURA 120
3.2.8.1 Objetos de datos para texto plano 123
3.2.8.2 Objetos de datos para mecanismos de seguridad 123
3.2.8.3 Objetos de datos para funciones auxiliares 124
3.2.9 CANALES LÓGICOS 125
3.3 COMANDOS DE TARJETAS INTELIGENTES 126
3.3.1 COMANDOS DE SELECCIÓN DE ARCHIVOS 131
3.3.2 COMANDOS READ AND WRITE 133
3.3.3 COMANDOS SEARCH 141
3.3.4 COMANDOS DE IDENTIFICACIÓN 143
3.3.5 COMANDOS DE ADMINISTRACIÓN DE ARCHIVOS 145
3.3.6 COMANDOS PARA PROTOCOLOS DE TRANSMISIÓN DE DATOS 146

Capítulo 4

Técnicas de seguridad

4.1 IDENTIFICACIÓN DEL USUARIO 149
4.1.1 IDENTIFICACIÓN DEL USUARIO USANDO UN PIN 152
4.1.1.1 Probabilidad de adivinar un PIN 154
4.1.1.2 Probando la autenticidad de un terminal 156
4.2 SEGURIDAD DE LA TARJETA INTELIGENTE 156
4.2.1 CLASIFICACIÓN DE ATAQUES Y ATACANTES 158
4.2.1.1 Clasificación de ataques 159
4.2.1.2 Clasificación de atacantes 161
4.2.1.3 Clasificación de la atracción de un ataque 162
4.3 CRIPTOGRAFÍA 164

4.3.1 ALGORITMOS CRIPTOGRÁFICOS SIMÉTRICOS 167
4.3.1.1 El Algoritmo DES 167
4.3.1.2 El Algoritmo IDEA 169
4.3.1.3 Modos de operación de los algoritmos de encripción orientados a bloques 170
4.3.1.4 Encripción múltiple 172
4.3.2 ALGORITMOS CRIPTOGRÁFICOS ASIMÉTRICOS 173
4.3.2.1 El Algoritmo RSA 174
4.4 ATAQUES Y CONTRAMEDIDAS DURANTE EL DESARROLLO 175
4.4.1 DESARROLLO DEL MICROCONTROLADOR DE LA TI 175
4.4.1.1 Protección: Criterio de Diseño 175
4.4.1.2 Protección: Números de CHIP ambiguos 176
4.4.2 DESARROLLO DEL SISTEMA OPERATIVO DE LA TI 177
4.4.2.1 Protección: Principios de desarrollo 177
4.4.2.2 Protección: Distribución del conocimiento 179
4.5 ATAQUES Y CONTRAMEDIDAS DURANTE EL PROCESO DE FABRICACIÓN 179
4.5.1 PROTECCIÓN: AUTENTICACIÓN EN LOS PASOS DE FABRICACIÓN 180
4.6 ATAQUES Y CONTRAMEDIDAS DURANTE EL USO DE LA TARJETA 181
4.6.1 ATAQUES AL NIVEL FÍSICO 181
4.6.1.1 Análisis estático de los microcontroladores de TIs 182
4.6.1.1.1 Protección: Tecnología del semiconductor 182
4.6.1.1.2 Protección: Diseño del chip 183
4.6.1.1.3 Protección: Buses sobre el chip 183
4.6.1.1.4 Protección: Diseño de la memoria 184
4.6.1.1.5 Protección: Capas protectoras (Escudos) 184
4.6.1.1.6 Protección: Scrambling de la Memoria 186
4.6.1.1.7 Protección: Encripción de la Memoria 186
4.6.1.2 Análisis dinámico de los microcontroladores de TIs 187
4.6.1.2.1 Protección: Monitoreo de la capa pasiva 187
4.6.1.2.2 Protección: Control de voltaje 188
4.6.1.2.3 Protección: Monitoreo de frecuencia 188

4.6.1.2.4 Protección: Scrambling de los buses 189

4.6.1.2.5 Análisis dinámico y defensa: Medición del consumo de energía del CPU 190

4.6.1.2.6 Análisis y defensa: Medición de la radiación electromagnética del CPU 193

4.6.2 ATAQUES AL NIVEL LÓGICO 193

4.6.2.1 Ataque y defensa: Tarjeta inteligente falsa 194

4.6.2.2 Ataque y defensa: Interrupción de energía 195

4.6.2.3 Ataque y defensa: Análisis de energía por comparación del PIN 196

4.6.2.4 Ataque y defensa: Análisis de tiempo por comparación del PIN 197

4.6.2.5 Ataque y defensa: Algoritmos criptográficos libres de ruido 198

4.6.2.6 Manipulación: Análisis diferencial por defecto (DFA) 199

4.6.2.7 Ataque y defensa: Perturbando el procesador 200

Capítulo 5

Aplicaciones de las tarjetas inteligentes

5.1 TARJETAS INTELIGENTES EN LOS SISTEMAS DE PAGO 202

5.1.1 TRANSACCIONES DE PAGO USANDO TARJETAS 203

5.2 TARJETAS INTELIGENTES EN TELECOMUNICACIONES 206

5.3 TARJETAS INTELIGENTES SIN CONTACTOS PARA VIAJES AÉREOS 209

5.4 TARJETAS DE SEGURO MÉDICO 211

5.5 SISTEMAS DE PEAJE ELECTRÓNICO 213

5.6 FIRMAS DIGITALES 216

5.7 OTRAS APLICACIONES

Capítulo 6

Diseño de un ejemplo de aplicación

6.1 OBJETIVOS 222

6.2 APLICACIÓN PROPUESTA 222

6.2.1 ANTECEDENTES 223

6.2.2 REQUERIMIENTOS 223

6.2.3 CARACTERÍSTICAS DEL PROBLEMA 224
6.2.4 HERRAMIENTAS UTILIZADAS EN LA SOLUCIÓN 226
6.2.4.1 Lector y Programador 226
6.2.4.2 Kit 226
6.2.4.3 Memory Card SLE4428 227
6.3 IMPLEMENTACIÓN DE LA APLICACIÓN 227
6.3.1 MAPA DE MEMORIA 227
6.3.2 ETAPA DE ESCRITURA 232
6.3.3 ETAPA DE LECTURA 237
6.3.4 PROPIEDADES DE SEGURIDAD 240
6.4 DETALLE DE COSTOS 242
CONCLUSIONES Y RECOMENDACIONES 244

Anexo A
Líneas de código

A.1 PROGRAMA DE ESCRITURA (Escritor.frm) A-1
A.2 PROGRAMA DE LECTURA (Lector.frm) A-9
A.3 MÓDULO (Modulo.bas) A-14

Anexo B
Tarjeta SLE 4428

B.1 CARACTERÍSTICAS B-1
B.2 DESCRIPCIÓN FUNCIONAL B-2
B.3 CARACTERÍSTICAS DE SEGURIDAD B-3
B.4 DATOS TÉCNICOS B-5

Anexo C
Librería ACSR 20

C.1 DESCRIPCIÓN C-1
C.2 VELOCIDAD DE COMUNICACIÓN C-2
C.3 API ACSR C-2
C.3.1 ESTRUCTURA DEL INTERFAZ DE DATOS C-3
C.3.1.1 AC_APDU C-3
C.3.1.2 AC_SESSION C-4

C.3.1.3 AC_INFO C-5
C.3.2 PROTOTIPOS DE FUNCIONES DE INTERFAZ C-6
C.3.2.1 AC_Open C-6
C.3.2.2 AC_Close C-7
C.3.2.3 AC_StartSession C-8
C.3.2.4 AC_EndSession C-9
C.3.2.5 AC_Exchange C-10
C.3.2.6 AC_GetInfo C-11
C.3.3 COMANDOS ACI C-12
C.3.3.1 ACI_Read C-13
C.3.3.2 ACI_Write C-13
C.3.3.3 ACI_Verify C-13

Anexo D

Lector y Programador ACR30

D.1 INTRODUCCIÓN D-1
D.2 TIPOS DE TARJETAS SOPORTADOS D-1
D.2.1 MEMORY CARDS (INTERFAZ SINCRÓNICA) D-2
D.2.2 TARJETAS BASADAS EN MICROCONTROLADORES (INTERFAZ ASINCRÓNICA) D-2
D.3 INTERFAZ DE LA TARJETA INTELIGENTE D-2
D.3.1 SUMINISTRO DE ENERGÍA A LA TARJETA INTELIGENTE (VCC) D-3
D.3.2 VOLTAJE DE PROGRAMACIÓN (VPP) D-3
D.3.3 SELECCIÓN DEL TIPO DE TARJETA D-3
D.3.4 INTERFAZ PARA TARJETAS BASADAS EN MICROCONTROLADOR D-4
D.3.5 PROTECCIÓN DE DESPLAZAMIENTO DE LA TARJETA D-4
D.4 SUMINISTRO DE ENERGÍA D-4
D.5 INTERFAZ SERIAL D-4
D.5.1 PARÁMETROS DE COMUNICACIÓN D-5
D.6 PROTOCOLO DE COMUNICACIÓN D-5

Índice de figuras

Tarjetas Inteligentes

Capítulo 1

Figura 1.1 Medidas y ubicación de los contactos en una TI (*Referencia:* [FREETUTORIALS]) 6

Figura 1.2 Distribución de los contactos (*Referencia:* [FREETUTORIALS]) 7

Figura 1.3 Clasificación de las Tarjetas Inteligentes, (*Referencia:* [ROOTSHELL]) 12

Figura 1.4 Arquitectura de una Memory Card (*Referencia:* [ROOTSHELL]) 14

Figura 1.5 Arquitectura simplificada de una Tarjeta con chip (*Referencia:* [ROOTSHELL]) 15

Figura 1.6 Tarjetas con contacto (*Referencia:* [NETHANDBOOK]) 16

Figura 1.7 Tarjeta sin contactos (*Referencia:* [NETHANDBOOK]) 17

Capítulo 2

Características físicas y propiedades eléctricas

Figura 2.1 Definición de las dimensiones de una tarjeta de formato ID-1. (*Referencia:* [WOLFGANG]) 31

Figura 2.2 El formato ID-1: Espesor: 0,76 mm ± 0,08 mm; radio de la esquina: 3,18 mm ± 0,30 mm. Las dimensiones mostradas indican el lado de la tarjeta excluyendo tolerancias. (*Referencia:* [WOLFGANG]) 32

Figura 2.3 Formato ID-000. (*Referencia:* [WOLFGANG]) 32

Figura 2.4 Formato ID-00. (*Referencia:* [WOLFGANG]) 33

Figura 2.5 Tamaño relativo de los formatos ID-1, ID-00 e ID-000. (*Referencia:* [WOLFGANG]) 33

Figura 2.6 Prueba de encorvamiento. (*Referencia:* [FREETUTORIALS]) 36

Figura 2.7 Prueba de torsión. (*Referencia:* [FREETUTORIALS]) 37

Figura 2.8 Prueba de resistencia a una descarga eléctrica estática. (*Referencia:* [FREETUTORIALS]) 37

Figura 2.9 Posición de los contactos relativa al cuerpo de la tarjeta. (*Referencia:* [WOLFGANG]) 38

Figura 2.10 Dimensiones mínimas de los contactos como se especifica en ISO/IEC 7816-2. (*Referencia:* [WOLFGANG]) 38

Figura 2.11 Diferentes formas de acomodar el chip, el campo de estampado en relieve y la banda magnética en una TI, de acuerdo con el estándar ISO 7816-2. (*Referencia:* [WOLFGANG]) 39

Figura 2.12 Fabricación del chip. (*Referencia:* [GUTHERY]) 40

Figura 2.13	Elementos de un módulo de Tarjeta Inteligente. (*Referencia:* [GUTHERY])	41
Figura 2.14	Inserción del módulo en el cuerpo de la tarjeta (*Referencia:* [GUTHERY])	41
Figura 2.15	Comparación de las 3 clases de voltaje y rangos de frecuencia fijados como se especifica en el estándar ISO/IEC 7816-3 y en la enmienda 1. (*Referencia:* [WOLFGANG])	48
Figura 2.16	Fotografía de la zona de contacto entre un cable y una almohadilla del microcontrolador de una TI, ampliada 1000 veces (*Referencia:* [WOLFGANG])	51
Figura 2.17	Consumo de corriente en el microcontrolador versus frecuencia de reloj en el modo de operación normal. El consumo de corriente en el modo de dormido con el reloj aplicado también es linealmente dependiente de la frecuencia de reloj y es aproximadamente 50 μA a 5 MHz, dependiendo del tipo de microcontrolador. (*Referencia:* [WOLFGANG])	54
Figura 2.18	Circuito del canal E/S entre el terminal y la tarjeta inteligente. (*Referencia:* [WOLFGANG])	62
Figura 2.19	Secuencias de activación y desactivación de las tarjetas inteligentes acorde al estándar ISO/IEC 7816-3, los intervalos t1 y t2 yacen en los rangos $400/f \leq t1 \leq 40.000/f$ y $t2 \leq 200/f$, $40.000/f \leq t3$. IR: Reset interno, AL: Reset asincrónico, SH: Reset sincrónico. (*Referencia:* [WOLFGANG])	65
Figura 2.20	Reset de la tarjeta cuando espera una respuesta sincrónica. (*Referencia:* [CARDWERK])	66
Figura 2.21	Acomodamiento básico de una memoria dividida en algunos bancos y el flujo de programa asociado. Este ejemplo muestra dos subrutinas localizadas en dos diferentes bancos de memoria que son llamadas desde una región común de memoria. Los números en paréntesis indican la secuencia de eventos en el proceso. (*Referencia:* [WOLFGANG])	70
Figura 2.22	Comparación del área de una sola celda-bit para varios tipos de memoria. (*Referencia:* [WOLFGANG])	72
Figura 2.23	Clasificación de memorias de microcontroladores de TIs, las PROM y EPROM ya no se usan en microcontroladores modernos, las FRAM están empezando a usarse en TIs. (*Referencia:* [WOLFGANG])	73
Figura 2.24	Estructura funcional básica de una ROM. (*Referencia:* [WOLFGANG])	74
Figura 2.25	Principio de operación de un circuito de una bomba de carga durante la carga (izquierda) y descarga (derecha). (*Referencia:* [WOLFGANG])	74

Capítulo 3

Sistemas operativos, protocolos de transmisión y comandos de las tarjetas inteligentes

Figura 3.1 Representación esquemática del desarrollo histórico de los COSs. (*Referencia:* [WOLFGANG]) 81

Figura 3.2 La figura de la izquierda muestra el particionamiento típico de una RAM de 256 bytes para un COS programado en assembler y la figura de la derecha la organización de la memoria de una RAM de 6 Kbytes para una TI de telecomunicaciones de alto rendimiento cuyo sistema operativo ha sido generado usando lenguaje C. (*Referencia:* [WOLFGANG]) 84

Figura 3.3 Ejemplo de la partición de una EEPROM para un COS. (*Referencia:* [WOLFGANG]) 86

Figura 3.4 Particionamiento simple de la región de archivos de un COS que soporta algunas aplicaciones independientes y físicamente separadas. (*Referencia:* [WOLFGANG]) 87

Figura 3.5 Clasificación de estructuras de archivos de TIs de acuerdo con el estándar ISO/IEC 7816-4. (*Referencia:* [WOLFGANG]) 90

Figura 3.6 Los varios tipos de archivos en un árbol de archivos de una TI. (*Referencia:* [WOLFGANG]) 91

Figura 3.7 El modelo OSI para comunicaciones entre el terminal y la TI. (*Referencia:* [WOLFGANG]) 95

Figura 3.8 Estructura básica y elementos de datos de la ATR. (*Referencia:* [WOLFGANG]) 98

Figura 3.9 Estructura de un comando con el protocolo T=0. (*Referencia:* [WOLFGANG]) 104

Figura 3.10 Un byte transmitido vía el interfaz E/S sin errores usando el protocolo T=0. (*Referencia:* [WOLFGANG]) 105

Figura 3.11 Un error de transmisión se indica en el protocolo T=0 por un nivel bajo en el interfaz E/S durante el tiempo de seguridad. (*Referencia:* [WOLFGANG]) 105

Figura 3.12 Una secuencia de comunicación típica T=0 con datos tanto en el comando como en la respuesta. (*Referencia:* [WOLFGANG]) 106

Figura 3.13 Estructura de un bloque transmitido protocolo T=1. (*Referencia:* [WOLFGANG]) 110

Figura 3.14 Procedimiento para la extensión del tiempo de espera. (*Referencia*: [WOLFGANG]) 113

Figura 3.15 Ejemplo del encadenamiento de bloques para la transmisión de datos desde el terminal a la tarjeta. (*Referencia:* [WOLFGANG]) 114

Figura 3.16 Estructura de una APDU comando. (*Referencia:* [WOLFGANG]) 117

Figura 4.6	Operación del algoritmo DES para encriptar datos. (*Referencia:* [WOLFGANG])	168
Figura 4.7	Encripción de datos con un algoritmo de encripción orientado al bloque en el modo ECB. (*Referencia:* [WOLFGANG])	171
Figura 4.8	Encripción de datos con un algoritmo de encripción orientado al bloque en el modo CBC. (*Referencia:* [WOLFGANG])	172
Figura 4.9	Principio de operación de encripción de datos usando triple DES. (*Referencia:* [WOLFGANG])	173
Figura 4.10	Encripción y deencripción usando un algoritmo de llave pública. (*Referencia:* [WOLFGANG])	174
Figura 4.11	Clasificación de los puntos de ataque sobre un microcontrolador de TIs al nivel físico. (*Referencia:* [WOLFGANG])	181
Figura 4.12	Comparación de un cabello humano con la estructura del semiconductor de un microcontrolador de TIs, magnificado 1000x. (*Referencia:* [WOLFGANG])	182
Figura 4.13	Fotografía de varias celdas RAM magnificada 3000x, sin protección de capas metálicas adicionales. La fotografía de abajo muestra los potenciales eléctricos de las mismas celdas RAM que han sido medidos usando un haz de electrones con el chip en operación. La distribución de ceros y unos en la RAM puede ser reconocida claramente. (*Referencia:* [WOLFGANG])	185
Figura 4.14	Comparación de una memoria convencional y una memoria desordenada usada en un microcontrolador de TI. (*Referencia:* [WOLFGANG])	186
Figura 4.15	Capa pasiva moderna magnificada 8000x. La separación de pistas es 4 μm y el principio de funcionamiento de este detector se basa en la medición de resistencia. (*Referencia:* [WOLFGANG])	187
Figura 4.16	Scrambling de buses en un microcontrolador de TIs, ilustrando el uso de un bus de datos de 8 bits entre el CPU y la RAM. Las líneas del bus de datos mostradas aquí representan los flujos de información. (*Referencia:* [WOLFGANG])	190
Figura 4.17	Diagrama del circuito de conexiones a un microcontrolador de TIs para hacer mediciones simples de corriente usando resistencias en serie. (*Referencia:* [WOLFGANG])	191
Figura 4.18	Vista de la parte posterior de un módulo de TI abierto, el chip de la izquierda es un microcontrolador PIC que está conectado a una memoria EEPROM a la derecha por cables y caminos. Este tipo de módulo es usado típicamente para clonar una TI y otros tipos de ataques sobre sistemas de TIs. (*Referencia:* [WOLFGANG])	195
Figura 4.19	Ejemplo del procedimiento de un nuevo balance en un monedero electrónico. (*Referencia:* [WOLFGANG])	195

Figura 4.20 Ejemplo de una rutina no robusta para el envío del contenido de un buffer de transmisión, la cual puede ser atacada exitosamente por la perturbación del procesador. (*Referencia:* [WOLFGANG]) 200

Capítulo 5

Aplicaciones de las tarjetas inteligentes

Figura 5.1 Esquema sencillo de un medio de pagos actual. (*Referencia:* [LIBURUTEGIA]) 203

Figura 5.2 Tarjetas inteligentes de crédito. (*Referencia:* [INGENICO]) 220

Figura 5.3 Esquema sencillo de una TI multiaplicación. (*Referencia:* [LIBURUTEGIA]) 221

Capítulo 6

Diseño de un ejemplo de aplicación

Figura 6.1 División de la memoria EEPROM de la Memory Card 228

Figura 6.2 Secuencia de Bytes escritos para cada registro y codificación de los campos Año, Nota, Período, Tipo de aprobación, Número de matrícula, Créditos, Número de Materia 229

Figura 6.3 Diagrama de flujo de la aplicación de escritura de curriculum en la tarjeta 232

Figura 6.4 Diagrama de flujo del bloque Escribir en tarjeta 233

Figura 6.4 Diagrama de flujo del bloque Escribir en tarjeta (Continuación) 234

Figura 6.5 Pantalla del interfaz del escritor con un número único válido ingresado 235

Figura 6.6 Mensaje de Error al ingresar número único no válido 235

Figura 6.7 Ejemplo de escritura satisfactoria de curriculum en la tarjeta 236

Figura 6.8 Diagrama de flujo de la aplicación de lectura de la tarjeta 238

Figura 6.8 Diagrama de flujo de la aplicación de lectura de la tarjeta (Continuación) 239

Figura 6.9 Pantalla de lectura exitosa 240

Anexo A

Líneas de código

Figura A.1 Formulario de Escritor A-16

Figura A.2 Formulario de Lector A-17

Anexo B

Tarjeta SLE 4428

Figura B.1 Diagrama de bloque de operación de SLE 4428 B-2

Anexo C
Librería ACSR 20

Figura C.1 Arquitectura de la librería ACSR20 C-1

Índice de tablas

Tarjetas Inteligentes

Capítulo 1

Tabla 1.1 Detalle histórico del desarrollo de la TI. (*Referencia:* [NETHANDBOOK]) 3

Tabla 1.2 Partes del estándar ISO 7816. (*Referencia:* [CARDWERK]) 6

Tabla 1.3 Descripción de los contactos. (*Referencia:* [CARDWERK]) 8

Capítulo 2

Características físicas y propiedades eléctricas

Tabla 2.1 Resumen de los tres parámetros eléctricos (voltaje, corriente y tasa de reloj) para los estándares internacionales e industriales más importantes. (*Referencia:* [WOLFGANG]) 49

Tabla 2.2 Características eléctricas de E/S bajo condiciones de operación normales. (*Referencia:* [CARDWERK]) 60

Tabla 2.3 Características eléctricas de VPP bajo condiciones de operación normales. (*Referencia:* [CARDWERK]) 60

Tabla 2.4 Características eléctricas de CLK bajo condiciones de operación normales. (*Referencia:* [CARDWERK]) 61

Tabla 2.5 Características eléctricas de RST bajo condiciones de operación normales. (*Referencia:* [CARDWERK]) 61

Tabla 2.6 Características eléctricas de VCC (Clase A) bajo condiciones de operación normales. (*Referencia:* [CARDWERK]) 61

Capítulo 3

Sistemas operativos, protocolos de transmisión y comandos de las tarjetas inteligentes

Tabla 3.1 Elementos de datos de la ATR y su asignación según el estándar ISO/IEC 7816-3. (*Referencia:* [WOLFGANG]) 97

Tabla 3.2 Resumen de protocolos de transmisión de acuerdo con ISO/IEC 7816-3. (*Referencia:* [WOLFGANG]) 100

Tabla 3.3 Tiempos de transmisión para algunos comandos típicos usando el protocolo T=0. (*Referencia:* [WOLFGANG]) 108

Tabla 3.4 Tiempos de transmisión para algunos comandos típicos usando el protocolo T=1. (*Referencia:* [WOLFGANG]) 110

Tabla 3.5 Etiquetas para objetos de datos de texto plano. (*Referencia:* [WOLFGANG]) 123

Tabla 3.6 Etiquetas para objetos de datos de autenticación. (*Referencia:* [WOLFGANG]) 123

Tabla 3.7 Etiquetas para objetos de datos de confidencialidad. (*Referencia:* [WOLFGANG]) 124

Tabla 3.8 Funcionalidad del comando SELECT FILE acorde el estándar ISO/IEC 7816-4. (*Referencia:* [WOLFGANG]) 132

Tabla 3.9 Funcionalidad del comando SEEK acorde el estándar GSM 11.11. (*Referencia:* [WOLFGANG]) 141

Capítulo 4

Técnicas de seguridad

Tabla 4.1 Criterio para determinar el esfuerzo y costo requerido para un ataque sobre los componentes de seguridad de hardware o software, basado en los prerrequisitos para un ataque. (*Referencia:* [WOLFGANG]) 163

Tabla 4.2 Tiempos de cómputo típicos para DES. (*Referencia:* [WOLFGANG]) 169

Tabla 4.3 Tiempos de cómputo típicos para IDEA. (*Referencia:* [WOLFGANG]) 170

Capítulo 6

Diseño de un ejemplo de aplicación

Tabla 6.1 Tamaño del campo correspondiente a datos a ser almacenados 231

Anexo B

Tarjeta SLE 4428

Tabla B.1 Contactos de la tarjeta SLE 4428 B-2

Tabla B.2 Datos técnicos de la tarjeta SLE 4428 B-5

Anexo C

Librería ACSR 20

Tabla C.1 Parámetros de la estructura AC_APDU C-4

Tabla C.2 Parámetros de la estructura AC_SESSION C-5

Tabla C.3 Parámetros de la estructura AC_INFO C-5

Tabla C.4 Parámetros de la función AC_Open C-7

Tabla C.5 Parámetros de la función AC_Close C-7

Tabla C.6 Parámetros de entrada de la función AC_StartSession C-8

Tabla C.7 Parámetros de salida de la función AC_StartSession C-9

Tabla C.8 Parámetros de entrada de la función AC_EndSession C-10

Tabla C.9 Parámetros de entrada de la función AC_Exchange C-10

Tabla C.10 Parámetros de salida de la función AC_Exchange C-11

Tabla C.11 Parámetros de entrada de la función AC_GetInfo C-11

Tabla C.12 Parámetros de salida de la función AC_GetInfo C-12

Tabla C.13 Estructura de parámetros del comando ACI_Read C-13
Tabla C.14 Estructura de parámetros del comando ACI_Write C-13
Tabla C.15 Estructura de parámetros del comando ACI_Verify C-14
Tabla C.16 Índice de llave C-14
Tabla C.17 Compatibilidad de tipos de tarjetas C-15
Tabla C.18 Códigos de Error C-16

Tarjetas Inteligentes

Capítulo 1

1.1 INTRODUCCIÓN

La tarjeta de plástico es un elemento cuyo formato proporciona gran facilidad de uso, fundamentalmente por su portabilidad, comodidad y sencillez de manejo, de ahí la amplísima difusión que ha tenido y está teniendo su utilización. En el desarrollo de toda la dinámica de las TIs[1] siempre han subyacido dos problemas básicos:

1. ¿Cómo identificar a un usuario de manera segura?, lo que equivale a decir cómo identificar de forma segura a todas las partes que intervienen en una transacción, siendo indiferente que su contenido sea económico o no. Los dispositivos tecnológicos que intervienen en una transacción representan en algunos casos a otros tantos individuos, los cuales deben mutuamente reconocerse de forma suficiente y segura.

2. ¿Cómo garantizar una transferencia electrónica en su totalidad?, procesando la información que en cualquier caso implica, sin que su contenido pueda ser manipulado ni simulado.

Para dar respuesta a ambas cuestiones se han venido desarrollando múltiples soluciones en los últimos cincuenta años. El último avance lo constituye la tecnología que denominamos TI. Una TI es, en síntesis, la incorporación operativa de un chip, es decir, un procesador de información, a una tarjeta de plástico similar en formato a las tarjetas financieras que son tan ampliamente conocidas y utilizadas en la actualidad.

[1] [TI] Tarjeta Inteligente

El origen remoto de la TI como medio de pago se sitúa en New York en el año 1.950. Un hombre de negocios, llamado Frank McNamara, invitó a comer a un restaurante a unos clientes suyos. Al ir a pagar la cuenta descubrió con sorpresa que no portaba dinero suficiente. Este hecho le hizo reflexionar, y llegó a la conclusión de que no era lógico que hombres de negocios como él no pudiesen disponer de su capacidad financiera, bien fuese en forma de saldos bancarios o en forma de capacidad crediticia, estando condicionados por el efectivo disponible que portasen en cada momento. Como consecuencia de sus reflexiones se puso en contacto con los 27 restaurantes de New York más frecuentados por hombres de negocios y promovió la emisión de la primera tarjeta de crédito: Diners Club (Club de Comidas). Al cabo de un año se habían emitido más de 42.000 tarjetas y se había facturado más de un millón de dólares por medio de dicha tarjeta.

La TI propiamente dicha (smart card) sitúa su origen más próximo en Francia a finales de la década de los 70. La Dirección General de Telecomunicaciones impulsó varios proyectos de innovación tecnológica: Videotex, Minitel, correo electrónico. El 21 de marzo de 1.979, al amparo de uno de ellos, las compañías CII Honeywell- BULL y Motorola, terminaron la primera TI, la cual portaba dos chips.

Como ya hemos comentado antes, las dos tecnologías ya citadas, la tarjeta Diners Club y la primera tarjeta portadora de chip, se enfrentaban a los mismos problemas básicos (identificación y seguridad). Asimismo, desde un principio ha estado claro que la respuesta a dichas cuestiones se tiene que conseguir mediante la conjunción de tres elementos:

MICROELECTRÓNICA, ya que la solución debe adecuarse a un formato de uso personal, perfectamente portátil, lo que implica soluciones de tecnología y diseño industrial insertas en el mundo de las soluciones de tamaño muy reducido.

PROCESO DE DATOS, la lectura, captura, proceso, gestión y envío de información, implica cuestiones de capacidad de proceso y almacenamiento de

información y de comunicaciones que deben ser resueltas de la forma más eficiente posible.

Tabla 1.1 Detalle histórico del desarrollo de la TI. (*Referencia:* [NETHANDBOOK])

Año	Suceso
1971	EE.UU., la patente para una tarjeta de crédito y los planes contables se archivan, teniendo capacidad para almacenar información y para responder a los sistemas controlados por computadora. Esta patente fue la base para una tarjeta con un chip incorporado.
1974	La idea de una tarjeta verdaderamente "inteligente" fue concebida por Roland Moreno, periodista francés que pensó en una tarjeta de crédito con un microprocesador.
1976/79	Una compañía de computadoras francesa (*CII Honeywell Bull*) desarrolla una tarjeta inteligente conjuntamente con *Motorola*.
1980	*El Laboratorio Nacional de Comunicaciones* lleva a cabo pruebas de la tarjeta inteligente en Francia.
1982	Francia comienza a utilizar las tarjetas en los puntos de uso y puntos de venta, las tarjetas son fabricadas por *France Telecom*.
1983	Deutsche Bundespost comienza a investigar la tecnología de la tarjeta inteligente y su aplicabilidad a los sistemas de teléfono en Alemania.
1983	EE.UU. El Departamento de Defensa prueba la tarjeta con "Circuito Integrado" para los sistemas de identificación militar.
1984/85	El gobierno francés apoya a la banca francesa, Groupement des Cartes Bancaires, en un proyecto para poner tarjetas inteligentes en ejecución en una escala total;16 millones de tarjetas se crean y se ponen en uso.
1986	France Telecom pone en el mercado 7 millones de tarjetas de teléfono para su uso.
1988	Primera tarjeta bancaria fabricada para Groupement des Cartes Bancaires.
1993	La investigación en los diversos sistemas de pago revela que el intercambio de la información entre las tarjetas y los lectores inteligentes requiere especificaciones. Aparece la primera tarjeta GSM-SIM.
1994	Especificaciones de EMV creadas para el efectivo electrónico.
1996	1.5 millones de tarjetas sin contactos son publicadas en Corea para pagar el autobús. Las TIs fueron utilizadas en los juegos olímpicos del verano de 1996 en Atlanta. Los usos no son solo para las telecomunicaciones y transacciones financieras, sino se extienden a otras industrias, poniendo la base para el uso futuro. Aparece la primer tarjeta RSA 1024 bits "criptoprocesador"
1997	Primera tarjeta de IC Java.
2000	Primera tarjeta de IC Windows 2000.
2000	Primera tarjeta de IC para SunRay Workstation.
Futuro	Tarjetas multi-plataforma y tarjetas de múltiples usos con nuevas aplicaciones (Ej. pago de alquiler, Intranet, incrementar el uso del Internet para las compras, etc.).

CRIPTOGRAFÍA, es preciso utilizar técnicas que posibiliten la máxima seguridad o inviolabilidad de la información gestionada durante todo el proceso, lo

que supone el desarrollo y utilización de elementos de hardware y software que den soporte a esta funcionalidad.

Los detalles históricos de la industria de la tarjeta inteligente se describen en la Tabla 1.1

1.2 ESTÁNDARES

La necesidad de que exista interoperabilidad mundial impone el establecimiento de estándares internacionales con respecto primero a las tarjetas como tales, y luego, al equipo que trabajaría con ellas y a los ambientes en los cuales serían utilizadas. El lugar para establecer tales estándares es la organización internacional de estándares (ISO).

1.2.1 ISO 7810 TARJETAS DE IDENTIFICACIÓN - CARACTERÍSTICAS FÍSICAS (1985)

Este estándar especifica las características físicas de las tarjetas de identificación incluyendo el material de la tarjeta, construcción y dimensiones nominales para tres tamaños de tarjetas (ID-1, ID-2 e ID -3). Es la tarjeta de identificación ID-1 la que forma la base de ISO 7816 -1 (que como veremos mas adelante es el estándar mas conocido referente a tarjetas inteligentes). Los parámetros principales de ISO 7810 son las dimensiones de la tarjeta de identificación ID-1 que se definen de: 85.6mm x 53.98mm x 0.76mm.

1.2.2 ISO 7811 TARJETAS DE IDENTIFICACIÓN - TÉCNICAS DE GRABACIÓN (1985)

Este estándar consta de cinco partes y cubre la especificación de la tarjeta de banda magnética y de la tarjeta con impresos en alto relieve.

1.2.3 ISO 7812 TARJETAS DE IDENTIFICACIÓN- SISTEMA DE NUMERACIÓN Y PROCEDIMIENTO DE REGISTRO PARA IDENTIFICACIÓN DE USUARIOS (1987)

Este estándar define el número de identificación de la tarjeta o PAN (número de cuenta primario) que consta de tres partes, el número identificador del emisor (IIN), el identificador individual de la cuenta y el dígito de cheque.

1.2.4 ISO 7813 TARJETAS DE IDENTIFICACIÓN - TARJETAS DE TRANSACCIÓN FINANCIERA (1987)

Este estándar define los requisitos para las tarjetas a ser utilizadas en transacciones financieras. Especifica las características físicas, disposición, técnicas de registro, sistema de numeración y procedimientos de registro. Se define tomando como referencia los estándares ISO 7810, ISO 7811 e ISO 7812. En detalle el estándar define las dimensiones físicas de la tarjeta de forma más exacta, como sigue,

. Ancho	85.46 mm	-	85.72 mm
. Altura	53.92 mm	-	54.03 mm
. Espesor	0.76 mm	±	0.08 mm

El espesor de la tarjeta es particularmente importante para los lectores de la TI debido a la construcción del mecanismo del conector de tarjeta. Este dispositivo consiste a menudo de un carro móvil que coloca la tarjeta bajo la cabeza del conector mientras la limpia si es necesario y aplica una acción de presión. La variación en el espesor o aún una concavidad leve de la tarjeta puede causar que la comunicación no se lleve a cavo.

1.2.5 ISO 7816 TARJETA INTELIGENTE

1.2.5.1 Descripción

Uno de los estándares mas conocido es el estándar ISO-7816. El estándar se ha establecido para describir las "tarjetas de identificación - tarjetas con circuito integrado con contactos". El estándar está formado de varias partes, las cuales son enumeradas en la Tabla 1.2.

Tabla 1.2 Partes del estándar ISO 7816. (*Referencia:* [CARDWERK])

ISO 7816	Descripción
ISO 7816-1	Características físicas.
ISO 7816-2	Dimensiones y localización de los contactos.
ISO 7816-3	Señales electrónicas y protocolos de la transmisión.
ISO 7816-4	Comandos para el intercambio Inter-industrial.
ISO 7816-5	Sistema de numeración y procedimiento del registro para los identificadores de aplicación.
ISO 7816-6	Elementos de datos Inter-industriales.

1.2.5.2 Parte 1: Características físicas de las tarjetas con circuito integrado

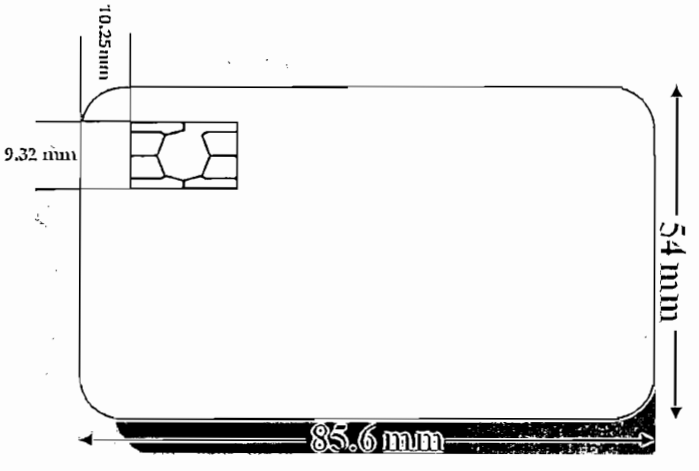

Figura 1.1 Medidas y ubicación de los contactos en una TI (*Referencia:* [FREETUTORIALS])

Aquí se describen las características físicas de las tarjetas. Incluye el establecimiento de límites para la exposición a un número de fenómenos electromagnéticos tales como rayos x, luz ultravioleta, campos electromagnéticos,

campos eléctricos estáticos, y temperatura ambiente que podrían afectar a la tarjeta.

Además el estándar ISO 7816-1 define las características de una tarjeta cuando es doblada. Este estándar debe cerciorarse de que las tarjetas plásticas con el chip incrustado sean fabricadas de manera que garantice el perfecto funcionamiento de la tarjeta durante el tiempo de vida previsto. Las conexiones entre los conectores superficiales y los pines de E/S de silicio incrustado deben estar fijas y resistir las tensiones mecánicas. La flexión y doblaje que soporta la tarjeta están estandardizados en ISO 7816-1 como se dijo antes. Esta parte de ISO 7816 es especialmente importante para los fabricantes de la tarjeta ya que son ellos los que eligen los materiales y establecen un proceso para encajar el circuito integrado en la tarjeta.

1.2.5.3 Parte 2: Dimensiones y ubicación de los contactos

La parte 2 de ISO 7816 define las dimensiones y la localización de los contactos. Esta parte estandariza el número, la función y la posición de los contactos eléctricos. La tarjeta con circuito integrado tiene 8 contactos eléctricos. Se nombran desde el contacto C1 hasta el C8. Sin embargo, no todos los contactos están conectados eléctricamente con el chip microprocesador incrustado y por lo tanto no todos se usan actualmente.

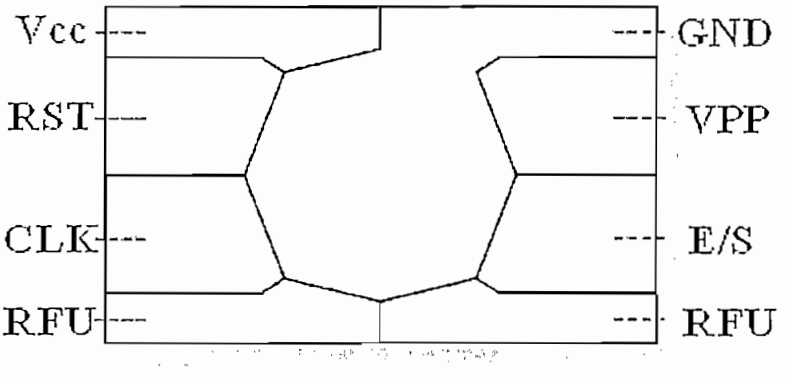

Figura 1.2 Distribución de los contactos (*Referencia:* [FREETUTORIALS])

Tabla 1.3 Descripción de los contactos. (*Referencia:* [CARDWERK])

Contacto	Designación	Descripción
C1	Vcc	Es la "toma de corriente" del chip. La energía la proporciona el dispositivo de hardware con el que la tarjeta interactúa en cada operación.
C2	RST	Es el mecanismo que pone en funcionamiento la interrelación entre una tarjeta inteligente y cualquier elemento externo adecuado con el que se ponga en contacto (TPV, ATM, TPS,...)
C3	CLK	Línea de señales de reloj a través de la cual una señal del reloj se puede proporcionar al microprocesador de la tarjeta inteligente para controlar la velocidad a la cual funciona y para proporcionar un marco común para la comunicación de datos entre el lector y la tarjeta inteligente.
C4	RFU	Reservado para uso futuro.
C5	GND	Línea de tierra que proporciona una tierra eléctrica común entre el lector y la tarjeta inteligente.
C6	Vpp	Conexión de energía de programación que proporciona una fuente separada de la corriente eléctrica (de la energía de funcionamiento) que se puede utilizar para programar la memoria permanente en el chip del microprocesador.
C7	E/S	Línea de la entrada-salida que proporciona un canal de comunicaciones half/duplex entre el lector y la tarjeta inteligente.
C8	RFU	Reservado para uso futuro.

1.2.5.4 Parte 3: Señales electrónicas y protocolos de transmisión.

La ISO 7816-3 comienza a describir la especificación de los aspectos verdaderamente "inteligentes" de la TI. Este estándar describe la relación entre la TI y el lector, como, un "esclavo" (la tarjeta) y el "amo" (el lector). Las comunicaciones son establecidas por el lector enviando señales a la TI a través de los contactos conocidos previamente y continuadas por la TI que responde según lo acordado.

En este estándar se definen los protocolos de transmisión (T=0) y (T=1) utilizados por las TIs, así como los niveles de voltaje que cada contacto admite, las secuencias de activación de las diferentes señales, y la función propiamente dicha de cada señal.

Se podría decir que la mayor parte de ISO 7816 parte 3 es importante para los fabricantes o desarrolladores de terminales que desean establecer una comunicación con una TI. A medida que uno se va adentrando en el estándar ISO

7816-3 se observa que guarda una estrecha relación con las líneas de código del software de entrada/salida. Dicho software permitiría a la tarjeta comunicarse con un puerto serial/paralelo/USB/PCMCIA de un microcontrolador o de una PC. En otra palabras trata sobre las señales electrónicas y protocolos que participan en el procesamiento de datos y la transmisión de información.

1.2.5.5 Parte 4: Comandos para el intercambio inter-industrial.

Esta parte de ISO 7816 especifica:

- el contenido de los mensajes, comandos y respuestas, transmitidos por el dispositivo de interfaz a la tarjeta y viceversa,
- la estructura y el contenido de los bytes históricos enviados por la tarjeta durante la solicitud de reinicio (ATR),
- la estructura de archivos y de datos, según lo visto en el interfaz al procesar los comandos para el intercambio inter-industrial,
- métodos de acceso a los archivos y datos en la tarjeta,
- métodos para la mensajería segura,
- métodos de acceso a los algoritmos procesados de la tarjeta. No describe estos algoritmos.

No cubre la implementación interna de la tarjeta y/o del mundo exterior. Permite además la estandarización de comandos ínter industriales y de arquitecturas de seguridad.

1.2.5.6 Parte 5: Sistema de numeración y procedimiento de registro de identificadores de aplicación.

ISO 7816-5 establece los estándares para los identificadores de Aplicación (AIDs). Un AID tiene dos partes. La primera es un identificador abastecedor de la aplicación registrada (RID) de cinco bytes que es únicamente para el vendedor. La segunda parte es un campo de longitud variable de hasta 11 bytes que el RID puede utilizar para identificar aplicaciones específicas.

1.2.5.7 Parte 6: Elementos de datos

La parte 6 del estándar ISO 7816 detalla el transporte físico de los datos del dispositivo y la transacción de datos, la solicitud de reinicio (ATR) y los protocolos de transmisión. Las especificaciones permiten dos protocolos de transmisión: protocolo del caracter (*T=0*) o protocolo del bloque (*T=1*). Una tarjeta puede soportar cualquiera pero no ambos. (nota: Algunos fabricantes de tarjetas no adhieren ni uno ni otro de estos protocolos. El protocolo de transmisión para tales tarjetas se describe como *T=14*).

El estándar ISO 7816 es indiscutiblemente el estándar de uso general más conocido y seguido de la tarjeta inteligente, pero de ninguna manera es el único. Hay estándares para el uso de tarjetas inteligentes en aspectos específicos tales como salud, transporte, actividades bancarias, comercio electrónico, e identidad. Y hay estándares para las nuevas clases de tarjetas inteligentes tales como las de proximidad y tarjetas inteligentes sin contacto. Puesto que una tarjeta inteligente es siempre parte de una tecnología de información más grande, debe siempre estar acorde a una amplia gama de estándares referentes al procesamiento de información tales como juegos de caracteres, códigos del país, representaciones monetarias, y criptografía. Puesto que muchas aplicaciones de la TI intersecan un número de preocupaciones gubernamentales tales como sistemas monetarios, identidad nacional, etc. existen estándares nacionales y regionales de la TI además de los estándares internacionales.

Existe también un número creciente de especificaciones de la TI publicadas por organizaciones tales como laboratorios gubernamentales, sociedades profesionales, asociaciones comerciales, instituciones académicas y firmas privadas no asociadas a los cuerpos de los estándares. Estas especificaciones no tienen ninguna fuerza con excepción de la fuerza del mercado de productos y de ideas. Mientras que las TIs se utilizan dentro de otras tecnologías tales como teléfonos portátiles, relojes, automóviles, y navegadores del Internet, podemos

tarjetas, aplicaciones y terminales. Un nuevo lanzamiento EMV2000 (EMV 4.0) está actualmente disponible.

1.3 CLASIFICACIÓN

Las **TIs** son los miembros más jóvenes de la familia de las tarjetas plásticas. Una **TI** se define como:

"Una tarjeta plástica, generalmente similar de tamaño y forma a una tarjeta de crédito y que contiene un microprocesador y memoria (que permite almacenar y procesar los datos)" esto, en conformidad con el estándar ISO 7816.

En un lenguaje más simple una **TI** se puede definir como una tarjeta con un microprocesador encajado en ella.

Aunque pueden ser clasificadas de acuerdo a toda una variedad de parámetros, en esta sección discutiremos únicamente la clasificación sobre la base de los componentes de la tarjeta, el interfaz de la tarjeta y los sistemas operativos de la tarjeta inteligente. Esta clasificación se representa mejor en la figura 1.3

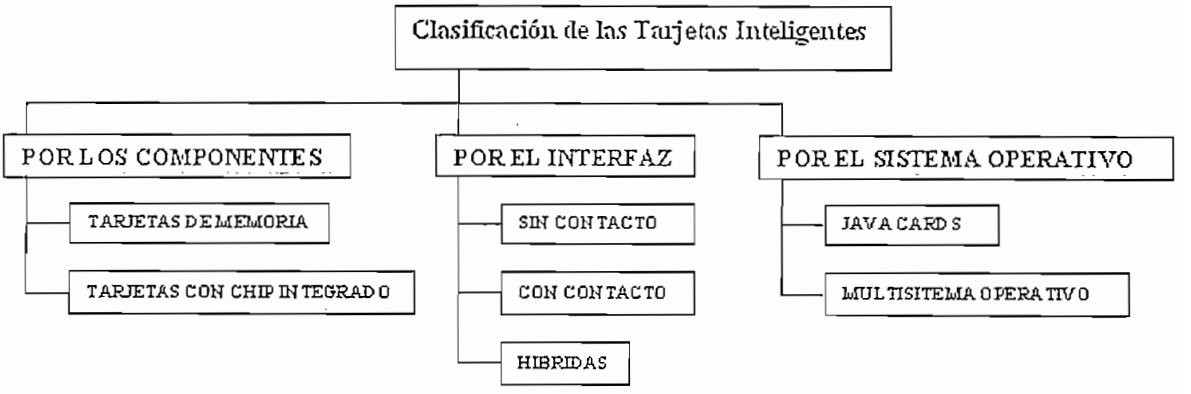

Figura 1.3 Clasificación de las Tarjetas Inteligentes, (*Referencia:* [ROOTSHELL])

1.3.1 CLASIFICACIÓN BASADA EN LOS COMPONENTES

Cuando se clasifica las TIs por los componentes que contienen, las mismas pueden ser divididas en dos categorías. Aquellas con un procesador se llaman Tarjetas con Chip, Chip Cards o Tarjetas con Microprocesador y aquellas sin chip se llaman Tarjetas de Memoria o Memory Cards.

1.3.1.1 Tarjetas de Memoria o Memory Cards

Contienen energía de procesamiento limitada, memoria limitada y pocas opciones de seguridad, además no pueden manejar archivos dinámicamente. Este tipo de tarjeta es muy similar a las tarjetas de banda magnética pero tiene más capacidad de memoria. Éstas son las tarjetas más comunes y las menos costosas. Contienen:

- EEPROM: Memoria ROM Programable y Reprogramable Eléctricamente. Es como un dispositivo de almacenamiento de datos en donde todos los datos de la aplicación son escritos. El tamaño típico de la EEPROM varía de 100 Bytes a 8 KB. Los datos de la EEPROM se pueden bloquear con un PIN[2] y varían generalmente con el tiempo. Por ejemplo, en una tarjeta del teléfono, la EEPROM podría almacenar el tiempo de charla.

- ROM: Memoria de sólo lectura. Almacena los datos que no cambian durante la vida de la tarjeta. Podría almacenar el número de tarjeta, el nombre del titular de tarjeta, etc.

Los controles lógicos de seguridad permiten el acceso a la memoria y habilitan la lectura y escritura en ella. Las regiones de la memoria son accesibles solamente después que se proporciona un código secreto. Este código se puede proporcionar por el lector de la TI o el titular de tarjeta. La arquitectura simplificada de una tarjeta de memoria se representa en la Figura 1.4.

[2] [PIN] Número de Identificación Personal

La tecnología simple de estas tarjetas les permite ser fabricadas con bajo costo. Estas tarjetas pueden almacenar datos desde 100 Bytes hasta 8 KB. Las Tarjetas de Memoria encuentran la aceptación amplia en el segmento de telefonía prepagada debido a su simplicidad. Otras áreas posibles donde pueden ser utilizadas incluyen las máquinas de venta, transporte, estacionamiento prepagado.

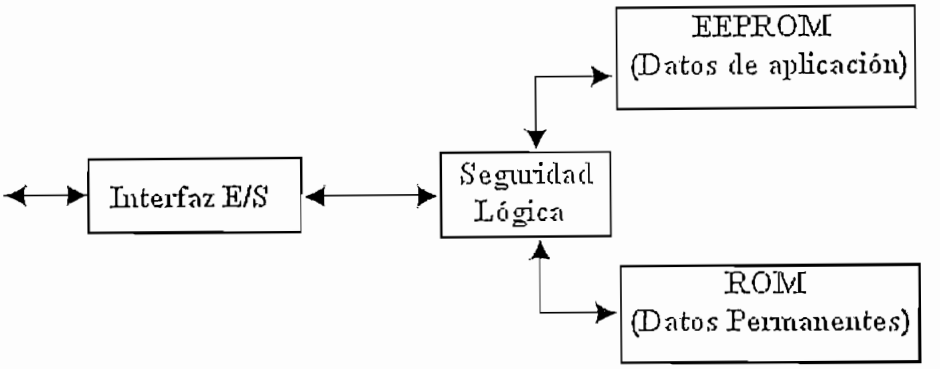

Figura 1.4 Arquitectura de una Memory Card (*Referencia:* [ROOTSHELL])

1.3.1.2 Tarjeta con chip

Como el nombre lo indica, éstas son las tarjetas que incorporan un microprocesador. Son las que técnicamente se pueden denominar **TIs**. Similar a una computadora con capacidad de procesar, almacenar y de asegurar la información. Además tiene capacidad de procesamiento de datos interno y cuentan con microprocesadores dedicados a funciones o aplicaciones específicas. Pueden administrar y controlar archivos y asignar memoria con el uso del sistema operativo de la tarjeta.

Estas tarjetas permiten que la información sea actualizada sin el reemplazo y el endoso de los productos y de los servicios de modo que proveen al usuario de comodidad, seguridad creciente y la facilidad de que algunas tarjetas, esencialmente, se combinen en una.

Los componentes importantes de una tarjeta con chip son:

- ROM: Memoria de solo lectura. La ROM contiene el sistema operativo de las tarjetas que también se conoce como *la máscara* de la tarjeta. Esto se escribe solamente una vez (generalmente durante la fase de la producción de la tarjeta). El tamaño de la ROM varía de algunos KBytes a 32 KB, dependiendo de cuál es el sistema operativo que está siendo utilizado por la tarjeta. Una vez que está escrita, no puede ser alterada.

- EEPROM: La EEPROM contiene los programas de aplicación de las tarjetas y los datos de la aplicación. Estos datos no son permanentes y a menudo se borran y se rescriben. Los tamaños típicos de EEPROM se encuentran en la gama de 2 KB a 32 KB.

- RAM: Memoria de Acceso Aleatorio. Ésta es una memoria volátil usada por el procesador para ejecutar las funciones deseadas. La memoria se borra siempre que se apague la energía. Suena sorprendente, pero, el tamaño típico de la RAM es alrededor de 256 bytes. Esto es porque la RAM ocupa el área máxima por byte y el área de una **TI** se restringe a 25mm cuadrados.

- CPU: Unidad Central de Procesamiento. Éste es el corazón de una tarjeta con chip. Es generalmente un microprocesador de 8 bits basado en arquitectura *CISC* con velocidades de reloj típicas de 5 MHz. Pero se está migrando lentamente hacia una arquitectura 32-bits debido a las tarjetas de Java. La CPU es responsable de realizar varias instrucciones.

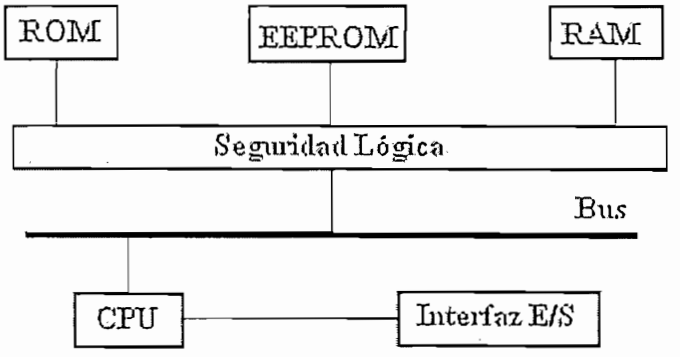

Figura 1.5 Arquitectura simplificada de una Tarjeta con chip (*Referencia:* [ROOTSHELL])

Las tarjetas con chip son más costosas que las tarjetas de memoria. Su coste se encuentra en el rango desde los $2 a $20 dependiendo de las características disponibles. Estas tarjetas pueden contener aplicaciones múltiples y proporcionar una seguridad robusta. Tales tarjetas se utilizan en el control de acceso, monederos electrónicos, crédito y otras tarjetas financieras, viajes, boletos y otras aplicaciones donde se requiere de alta seguridad. Una versión simplificada de la arquitectura interna de una tarjeta con chip se muestra en la Figura 1.5.

1.3.2 CLASIFICACIÓN BASADA EN EL INTERFAZ

Las TIs también se clasifican en base al método de comunicación y transferencia de datos con el dispositivo lector. De acuerdo con este criterio, las TIs son clasificadas como *tarjetas de contacto, tarjetas sin contacto, tarjetas combi y tarjetas híbridas.* Las tarjetas de contacto tienen que insertarse en el lector mientras que las tarjetas sin contacto son accionadas por una señal de radiofrecuencia y no requiere la inserción en un lector. Las Tarjetas Combi, así como las Tarjetas híbridas, se pueden accionar por la inserción o por una señal de radiofrecuencia.

1.3.2.1 Tarjetas de Contacto

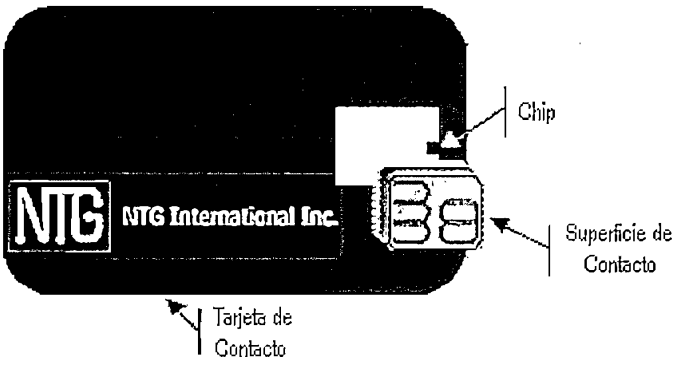

Figura 1.6 Tarjetas con contacto (*Referencia:* [NETHANDBOOK])

Estas tarjetas requieren la inserción en el lector para ser accionadas. Cada tarjeta contiene de 6 a 8 contactos plateados que están en contacto físico con el

lector. El contacto físico puede ser establecido *resbalando* o *aterrizando.* La tarjeta recibe la energía del lector por medio de los contactos destinados para esto. La designación y función de los contactos se detallaron anteriormente.

Las tarjetas de contacto tienen ciertas limitaciones. Con el tiempo, estos contactos tienden a despegarse. Las descargas electrostáticas, debido al contacto incorrecto pueden dañar los circuitos. Los portadores de la tarjeta algunas veces sacan las tarjetas del lector antes de que se termine la transacción, esto conduce a lo que se conoce como rasgado de la tarjeta. La manipulación ruda y tensiones durante la inserción de la tarjeta conducen al daño de la misma.

1.3.2.2 Tarjetas sin contacto

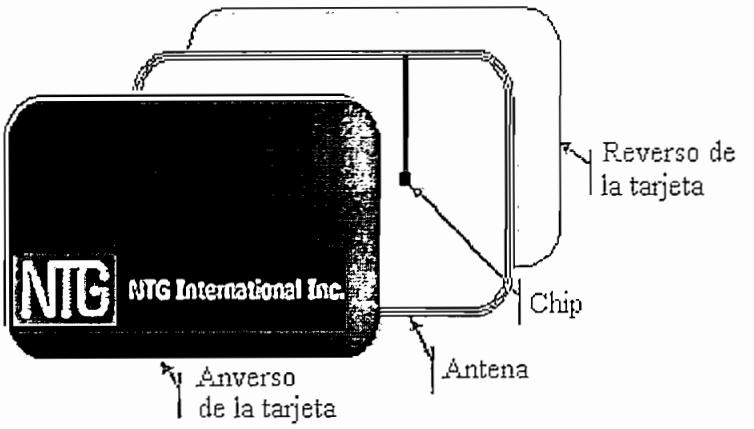

Figura 1.7 Tarjeta sin contactos (*Referencia:* [NETHANDBOOK])

Las tarjetas sin contacto no requieren la inserción en el lector. Apenas tienen que ser pasadas cerca de una antena para que la transacción sea realizada. La distancia de la lectura varía de unos pocos centímetros a unos 50 cm. Como no hay contacto, estas tarjetas solucionan la mayoría de las limitaciones enumeradas para las tarjetas de contacto. Tales tarjetas se utilizan a menudo en los lugares en donde la transacción tiene que ser realizada muy rápidamente. Por ejemplo: tránsito masivo, pago de peajes, etc. Las tarjetas sin contacto son más costosas comparadas a las tarjetas con contacto. Pero también tienen una mayor vida y son más confiables.

1.3.2.3 Tarjetas Combi y Tarjetas Híbridas

Las Tarjetas Combi al igual que las Tarjetas Híbridas son aquellas que cuentan tanto con el interfaz para tarjetas de contacto como para tarjetas sin contacto, facilitando así su utilización. Por ejemplo, una tarjeta de contacto podría incrustarse dentro de una slot que tiene una batería y una antena y puede comunicarse con un lector de tarjetas sin contacto, creando de esta forma una tarjeta capaz de comunicarse con cualquiera de los dos lectores. Otras tarjetas podrían ser más simples con dos chips, uno para los lectores de tarjetas de contacto y otro para los lectores de tarjetas sin contactos, estas últimas son las llamadas Tarjetas Híbridas, conviene señalar que ambos chips no se comunican dentro de la tarjeta. Ahora, una Tarjeta Combi es aquella que posee un solo chip que lleva ambos interfaces incluidos y por tanto las aplicaciones del interfaz sin contacto pueden comunicarse con las aplicaciones del interfaz de contacto. En ambos tipos de tarjetas el interfaz sin contacto es usado para aplicaciones que requieren rápidos tiempos de transacción y el interfaz de contacto es usado para esas aplicaciones que requieren la mayor seguridad posible.

1.3.3 CLASIFICACIÓN BASADA EN LOS SISTEMAS OPERATIVOS

Las tarjetas inteligentes también se clasifican en base de su sistema operativo. Hoy en día existen muchos sistemas operativos disponibles en el mercado para la tarjeta inteligente, los principales son:

1. MultOS
2. JavaCard
3. Cyberflex
4. StarCOS
5. MFC

Los sistemas operativos de la tarjeta inteligente o COS como se llaman comúnmente, se colocan en la ROM y ocupan generalmente menos de 16 KB.

Los COS se encargan:

- Del direccionamiento y manipulación de los archivos,

- de la administración de la memoria, y
- de los protocolos de transmisión de datos.

1.4 APLICACIONES

Esta sección describe porqué las organizaciones deben considerar el usar TIs. Se explican las ventajas dominantes de la tecnología de la TI y se precisan los obstáculos actuales que existen para la aceptación de la tecnología de la misma. Finalmente damos los ejemplos de donde las TIs se están utilizando hoy.

1.4.1 LA TARJETA INTELIGENTE EN EL ENTORNO DE LAS ENTIDADES FINANCIERAS

Hoy en día la utilización de tarjetas de plástico se produce fundamentalmente en el entorno financiero debido a la intensa actividad emisora desarrollada por prácticamente todas las entidades financieras que se dedican a la banca de particulares, actividad especialmente desarrollada en los últimos quince años. La finalidad fundamental con que nacieron las tarjetas financieras en dicho entorno fue la de servir tanto como de medios de pago (tarjetas de crédito primero, y de débito después) como de medio de dispensación de dinero efectivo (activación de ATMs - cajeros automáticos).

La evolución de los sistemas financieros y de las políticas de marketing de empresas de otros sectores de actividad, ha llevado a que alcance su adecuada dimensión otra importante función de las tarjetas desde el punto de vista de sus emisores: la de ser vehículo activo para la relación de clientela. En el momento presente las entidades financieras emiten tres tipos básicos de tarjetas:

1. Tarjeta de CRÉDITO: los pagos realizados con las mismas se cobran inmediatamente por los vendedores y se pagan al cabo de cierto plazo por los compradores. Esta forma de operar se financia mediante el pago de una comisión que cubre, además, otras funciones. Incorporan además la posibilidad de financiar

la operación a un plazo mayor que el del mero diferimiento del pago, es decir, pago aplazado y fraccionado. Estas tarjetas fueron las primeras en funcionar ya que básicamente eran off-line (aunque con autorización previa del emisor o del operador)

2. Tarjeta de DÉBITO: suponen un importante adelanto tecnológico respecto de las anteriores, ya que se trata de pagos realizados on-line contra el saldo efectivo existente en un depósito bancario, bien sea para disposición de efectivo en ATM o bien sea para transferencia automática entre cuentas bancarias en una transacción comercial.

3. Tarjeta MONEDERO ELECTRÓNICO: viene a ser el primer desarrollo basado en la tecnología de la TI. Dada su capacidad de almacenamiento y procesamiento de información, la memoria con que cuenta la tarjeta se carga con valor monetario descargado de un depósito bancario. Este valor monetario es exactamente igual que dinero efectivo: la única diferencia consiste en que en los billetes el valor monetario está materializado en papel y en el monedero se materializa digitalmente. El valor monetario incorporado en la tarjeta se va actualizando con cada transacción que se realiza, y pasa de una memoria (la de la tarjeta monedero) a otra memoria (la del terminal del comerciante) o de nuevo, a un depósito bancario o a dinero en billetes (a través de un cajero automático). Para hacernos una idea del impacto que más adelante puede suponer la introducción de la tarjeta inteligente en el entorno financiero, debemos tratar de entender de una forma sencilla el esquema operativo del entorno de medios de pago.

En una descripción sencilla de los modernos sistemas de pagos podemos distinguir cuatro tipos de elementos:

1. Activadores: se denomina así a las tarjetas, ya que por un lado, son los elementos que ponen en funcionamiento (activan) al hardware con el que se relacionan y por otro lado, con su utilización a través de dichos equipos

(interfaces) activan la relación de clientela con el emisor haciendo dicha relación operativa.

2. **Interfaces:** es el hardware que pone en contacto al usuario del activador con su intermediario financiero, bien sea directamente o bien sea a través de un operador de medios de pago.

3. **Intermediarios:** pueden ser vistos desde una perspectiva "personal" (describiendo quienes son), pero es más adecuada su visión como impulsores de grandes redes de distribución y de gestión operativa, ya que, en última instancia, posibilitan por adición la interrelación entre todos los agentes finales del mercado "real" (compradores y vendedores).

4. **Usuarios finales:** son los clientes particulares (en aras de simplificar, bien sean personas físicas o jurídicas) y los comerciantes o vendedores al detalle.

En una visión global de este esquema hay que observar el papel central de los intermediarios, muy especialmente de las entidades financieras, que posibilitan, mediante la emisión de tarjetas y la creación de infraestructuras interconectadas de interfaces y operadores, la operatoria final de los agentes no financieros. Las entidades financieras, con toda su infraestructura de medios de pago ya existente en el mercado, van a ser unos de los principales impulsores de la introducción de la tarjeta inteligente, activando principalmente su aplicación como monedero electrónico.

En el entorno financiero actual se está produciendo una importante transformación de los medios ó interfaces de relación con los clientes. Esta transformación viene impulsada, en primer lugar, por los avances tecnológicos, que posibilitan la existencia de dispositivos (cajeros automáticos, terminales punto de venta, terminales de autoservicio...) cada vez más eficientes y sofisticados. Al mismo tiempo se va produciendo un movimiento de integración de la tecnología (fundamentalmente vía PCs) en la vida de los ciudadanos, tanto en los entornos de trabajo como en los hogares. Esta integración está incidiendo en los hábitos sociales, que se están modificando de forma pausada pero segura, y dicha modificación conlleva la demanda de nuevas funcionalidades y *servicios*, que, a su vez, implican la necesidad de nuevos interfaces. De ahí el cambio funcional

que ostensiblemente están sufriendo las sucursales bancarias, así como el establecimiento de nuevas formas de interrelación entre las entidades financieras y sus clientes (banca telefónica, banca electrónica, banca en Internet...).

Este entorno, cada vez más sofisticado tecnológicamente (lo que no implica que tenga que ser operativamente más complejo), es caldo de cultivo perfecto para la introducción de la tecnología de la TI. Ya que, como se ha explicado antes, la tarjeta inteligente posee una mayor capacidad de almacenar información, puede gestionar activamente tanto la información que posee como la que recibe de otro dispositivo y, muy importante, se puede insertar en procedimientos operativos de muy alta seguridad, nos hallamos ante un elemento que puede perfectamente incorporarse junto con otros dispositivos en redes operativas de alto valor añadido. Por ello cada vez más nos vamos a ir encontrando en un entorno en el que todo tipo de dispositivos (ATMs, TPSs, TPVs, PCs, TV digital, teléfonos particulares y públicos, vehículos...) tendrán incorporado un lector/grabador de tarjetas inteligentes.

Por lo anterior, cada vez serán más los entornos en que se podrá utilizar la TI, incluso en algunos entornos (Ej. Internet) podrá llegar a ser obligatorio para la realización de determinadas operaciones (accesos a bases de datos, comercio electrónico), ya que las tarjetas inteligentes son el vehículo idóneo como soporte de certificados SET (Secure Electronic Transaction), que posibilitarán la realización de todo tipo de transacciones electrónicas de forma totalmente segura.

Las tarjetas financieras con el chip incorporado posiblemente evolucionarán hacia un único "plástico", en el que podrán convivir y se podrán utilizar "a la carta" los diferentes tipos de sistemas de pago (crédito, débito y monedero) que conocemos en la actualidad. Y tal vez veamos nuevos servicios financieros incorporados en dichas tarjetas, desarrollados a partir de la gran potencialidad que proporciona la TI.

Por todo ello, por lo menos, podemos prever sin dificultad la integración de esta tecnología en los actuales sistemas de medios de pago. Las TIs no van a

aportar nada novedoso en sí mismo al entorno de servicios y productos de las entidades financieras pero sí van a posibilitar una materialización de la relación de clientela financiera mucho más conveniente y satisfactoria para los clientes, que van a poder disponer de forma más efectiva de aquellos productos y servicios que atiendan adecuadamente sus necesidades. Y, además, posibilitarán la disponibilidad de otros servicios adicionales de mayor valor añadido que los servicios auxiliares que hoy en día se anexan a las tarjetas financieras de banda magnética.

Por todo lo señalado, se puede prever que la TI se va a constituir en el próximo soporte estándar del sistema de medios de pago. Y, dada la velocidad actual de evolución de acontecimientos y tecnologías, dicha previsión se debe materializar en un futuro muy cercano.

1.4.2 LA TARJETA INTELIGENTE EN OTROS ENTORNOS

Es un hecho reiteradamente mencionado que el desarrollo de las tecnologías de la información y de las telecomunicaciones y el abaratamiento del coste de almacenamiento de información están propulsando de forma definitiva un conjunto de modificaciones que podemos calificar como la revolución de la información. Hoy en día la actividad de muchos sectores se está reordenando y reconduciendo para centrarse en la gestión de información. Aquellos sectores que producen bienes materiales utilizan intensivamente diferentes tipos de información (de producción, de gestión, comercial...) como herramientas decisivas para el éxito en su actividad. En muchos sentidos estamos viviendo en la sociedad de la información. En este contexto, la tarjeta inteligente es un elemento que, dadas sus propias características, se puede insertar de forma natural en toda la cadena de generación y gestión de información en múltiples sectores de actividad: allí donde se pueda realizar un proceso con gestión de información puede insertarse una tarjeta inteligente.

A continuación se presenta una visión general y rápida de *cómo la* tarjeta inteligente puede ser utilizada en múltiples entornos diferentes. Precisamente por

los desarrollos que se están realizando en materia de seguridad de transacciones, una primera función genérica para la que puede ser utilizada la tarjeta inteligente es como herramienta de identificación. Cuando se hace referencia a identificación se está mencionando todo tipo de identificaciones personales que tengan utilidad tanto para el acceso a espacios físicos (el propio domicilio, una biblioteca, un área de uso restringido, etc.) como para el acceso a espacios virtuales (una zona de una base de datos, un conjunto de servicios prestados por medio de Internet). Sin dejar volar demasiado la imaginación podemos pensar en que todo tipo de carnés (carné de identidad, de socio de un club, de conducir, etc.) y de permisos (licencia temporal para estacionar en una zona determinada de una ciudad, etc.) pueden perfectamente incorporarse en una tarjeta inteligente. La combinación de la tecnología de la tarjeta inteligente con los procesos de seguridad de operaciones que se están desarrollando (RSA, SET, etc.) o que se puedan desarrollar, proporcionaría una herramienta de gran potencial.

La telefonía, tanto la telefonía básica como la GSM, constituye un terreno perfectamente susceptible de incorporar utilizaciones de tarjeta inteligente. Estos usos telefónicos se centran principalmente en las denominadas tarjetas de prepago (adquisición del derecho a utilizar comunicaciones telefónicas por un determinado valor económico) y en la identificación del usuario para la utilización de dispositivos. No es difícil pensar en que la TI utilizada en terminales telefónicos debidamente habilitados se pueda convertir en elemento de identificación de su titular y, consiguientemente, legitimarle para realizar determinadas operaciones; o que la propia tarjeta pueda enviar y recibir información por este conducto.

En materia de comercio electrónico la TI puede llegar a adquirir una especial relevancia por varias razones. La primera es que, como procesador de información, tal como hemos reiterado ya varias veces, puede intervenir activamente en procesos de seguridad de transacciones, no solo identificando a su titular, sino también colaborando para que su titular tenga la seguridad de que se está poniendo en contacto con otro usuario legítimo. Asimismo puede almacenar valor económico (dinero) y posibilitar la adquisición de bienes y servicios a través de redes de información, transfiriendo o recibiendo valor

económico con la otra parte de cualquier transacción. Y al hablar de comercio electrónico no debemos pensar nada más en los actuales centros de compras (malls) que existen en Internet, también el "pay per view" televisivo y todas sus posibles implicaciones, la consulta a bases de datos no gratuitas, es decir, hay que pensar en cualquier transacción de cualquier tipo de mercancía (en un sentido muy amplio) que se pueda imaginar.

El entorno de los transportes públicos es uno de los que más experimentos está concentrando, ya que se entiende que es uno de los terrenos que más rápidamente puede contribuir a la masificación de las TIs. En este entorno concreto se puede materializar por medio de TIs toda la gestión de bonos de viajes, precios especiales para determinados colectivos (descuentos para pensionistas), gestión de transbordos dentro de una misma línea e incluso entre líneas de diferentes compañías.

En las empresas de transportes se ha suscitado un gran interés por esta tecnología, ya que las posibilidades de gestión que proporciona en tiempo real de información de viajeros, combinadas con la utilización de otros medios técnicos, abren las puertas a una gestión realmente eficiente de las flotas de vehículos.

En todo lo referente a salud también la TI puede realizar una interesante aportación. Ya se han realizado varias experiencias tendientes a integrar el historial clínico de las personas en TIs. Las ventajas que puede aportar una utilidad semejante pueden ser muy importantes, ya que, especialmente en situaciones de urgencias clínicas, posibilitarían conocer rápidamente todas las vicisitudes médicas de cada persona de cara a realizar actuaciones médicas de mínimos riesgos para la vida del individuo. En una línea similar se ha avanzado en funciones de historial farmacológico, que posibilitaría el conocimiento inmediato de las ingestas más recientes de medicamentos, posibles alergias o contraindicaciones experimentadas por un individuo concreto.

Asimismo se están desarrollando experiencias de gestión hospitalaria, consistentes en que toda la gestión de las personas ingresadas en centros

hospitalarios o clínicas se soporte en TIs. Dicha tarjeta no solo contiene la información de registro del paciente, sino también información propiamente sanitaria referente a las visitas médicas, tratamiento prescrito y administrado, e información de gestión del centro, como comidas administradas, uso de teléfono y televisión.

Los hospitales, como tantos otros ejemplos de centros de actividades "cerradas" que se nos pueden ocurrir (universidades, clubes deportivos, etc.) se pueden convertir en verdaderos laboratorios de desarrollo de experiencias con TIs.

1.4.3 ¿PORQUÉ CONSIDERAR LAS TARJETAS INTELIGENTES?

La práctica indica lo siguiente a las organizaciones que consideran la incorporación de la tecnología de la TI en sus operaciones:

SI

- Un expediente portátil de una o más aplicaciones es necesario o deseable.
- Los expedientes probablemente requieren la puesta al día en un cierto plazo.
- Los expedientes se interconectarán con más de un sistema automatizado.
- La seguridad y confidencialidad de los expedientes son importantes.

ENTONCES

La TI es una solución factible para hacer la transferencia y el procesamiento de datos más eficiente y seguro.

1.4.4 VENTAJAS DE LAS TARJETAS INTELIGENTES

Las ventajas dominantes de la tecnología de la TI incluyen:

- La capacidad proporcionada por el microprocesador y la capacidad de almacenamiento de datos altamente seguro, combinado con el procesamiento de datos interno.
- Adherencia a los estándares internacionales, asegurando de esta manera múltiples fuentes de proveedores y precios competitivos.
- Resultados satisfactorios del uso de TIs en aplicaciones reales en todo mundo.
- Durabilidad y un tiempo de vida estimado largo (garantizados por el vendedor para alrededor de 10.000 lecturas/escrituras antes de alguna falla).
- Sistemas operativos que soportan aplicaciones múltiples y aseguran almacenaje de datos independiente en una sola tarjeta.

1.4.5 BARRERAS PARA LA ACEPTACIÓN DE LAS TARJETAS INTELIGENTES.

Los obstáculos actuales para la aceptación de la tecnología de la TI incluyen:

- Un coste relativamente más alto de las TIs con respecto a tarjetas magnéticas. (la diferencia se da inicialmente en el costo de las dos tecnologías, sin embargo, disminuye significativamente cuando las diferencias en vida prevista y las capacidades --particularmente en términos de soportar aplicaciones múltiples y de esa forma producir el reparto de gastos entre abastecedores de la aplicación-- son consideradas.)
- La actual carencia de infraestructura que soporte la tarjeta inteligente, haciendo necesaria la readecuación del equipo tal como máquinas de venta, ATMs y teléfonos.
- La naturaleza propietaria del sistema operativo del chip. El consumidor debe poseer buenos conocimientos técnicos para seleccionar la tarjeta más apropiada para la aplicación deseada.

- La carencia de los estándares para asegurar interoperabilidad entre los varios programas de la TI.
- Las ediciones legales y de política sin resolver, tales como las relacionadas con la privacidad y la confidencialidad o con las leyes de protección al consumidor.

A modo de conclusión podemos decir que la tarjeta inteligente es un elemento que tiene grandes visos de convertirse en un elemento cotidiano en nuestras vidas en un plazo de tiempo relativamente corto.

Capítulo 2

Características físicas y propiedades eléctricas

2.1 CARACTERÍSTICAS FÍSICAS

Muchos estándares relacionados con las propiedades físicas de la tarjeta no son específicos para la TI, pero se aplican igual tanto como con las tarjetas de banda magnética y aquellas grabadas en relieve. Uno de estos estándares es el ISO/IEC 7810: Características Físicas de la Tarjeta de Identificación, mencionado anteriormente, que define como su nombre lo indica las características físicas nominales para tres tipos de las tarjetas de identificación, etiquetados ID-1, ID-2, e ID-3. El tipo ID-1 se ocupa del tamaño y de la forma generalmente aceptados de una "tarjeta de crédito" o de la "tarjeta inteligente". Los tipos de tarjeta ID-2 e ID-3 son tamaños simplemente más grandes, pero con las mismas características físicas que las tarjetas del tipo ID-1.

Al sostener una tarjeta en la mano, lo primero que uno aprecia es su formato. Después de esto, podemos notar que posee un conjunto de contactos, aunque las TISC[1] no tienen ningún interfaz eléctrico visible. La siguiente característica que podríamos notar es la presencia de una banda magnética, texto grabado en alto relieve o un holograma. Todos estos componentes forman parte de las características físicas de una TI. La mayoría de las características físicas de las TI son en la actualidad puramente mecánicas en esencia, tal como el formato de la tarjeta y su resistencia a ser doblada o torcida. Esas características son familiares para cada usuario por experiencia personal. Pero en la práctica, propiedades como la sensibilidad a la temperatura, a la luz y al maltrato son también importantes.

La interacción entre el chip incrustado y el cuerpo de la tarjeta debe ser también considerada, puesto que únicamente la combinación de estos dos componentes hace funcional a una tarjeta. Estos dos componentes deben

[1] [TISC] Tarjeta Inteligente sin Contactos

individual y colectivamente conocer todos los requerimientos relevantes, de otro modo al momento de usarlas sería de esperarse altas tasas de error.

2.1.1 FORMATOS DE LA TI

Las tarjetas pequeñas con las dimensiones típicas de una TI de 85.6 mm por 54 mm han permanecido en uso por mucho tiempo. Casi todas las TIs son producidas en este formato, el cual ciertamente es el más familiar. La misma está diseñada en base a la ID-1 y su tamaño está especificado en el estándar ISO 7810. Este estándar fue creado en 1985 y por tanto no fue creado para las TIs que conocemos hoy, como se puede observar fácilmente desde la abreviatura 'ID', que simboliza 'identificación'. Este estándar simplemente describe una tarjeta plástica grabada en relieve con una banda magnética cuyo fin es ser usada para la identificación de una persona. La presencia de un chip y la localización de sus contactos sobre la tarjeta fueron definidas solamente algunos años mas tarde en otros estándares.

Con la variedad de tarjetas disponibles hoy en día, las cuales son usadas para todos los posibles propósitos y tienen un amplio rango de dimensiones, es a menudo difícil determinar si una tarjeta particular es realmente una tarjeta ID-1. A más del chip incrustado, una de las mejores características de identificación es el espesor de la tarjeta. Si mide 0.76 mm y la tarjeta contiene un microcontrolador, ésta puede ser considerada como una TI según el estándar ISO.

El formato convencional de una tarjeta ID-1 tiene la ventaja de ser muy fácil de manipular. El formato de la tarjeta está especificado de tal forma que no sea muy larga, como para poder llevarlo en una billetera, ni tan pequeño como para perderla fácilmente. En suma, la flexibilidad de las tarjetas las hace más convenientes que un objeto rígido.

No obstante, este formato no siempre está acorde con las necesidades de la miniaturización moderna. Algunos teléfonos móviles pesan únicamente 200 g y no son más grandes que un paquete de pañuelos. Por tanto fue necesario

- Rectángulo externo largo: 85,72 mm (3,375 pulgadas)
  ancho: 54,03 mm (2,127 pulgadas)
- Rectángulo interno largo: 85,46 mm (3,365 pulgadas)
  ancho: 53,92 mm (2,123 pulgadas)

El espesor debe ser 0,76 mm (0,03 pulgadas), con una tolerancia de ± 0,08 mm (± 0,003 pulgadas). Basados en estas definiciones, las dimensiones de una tarjeta ID-1 pueden ser representadas como se muestra en la Figura 2.2.

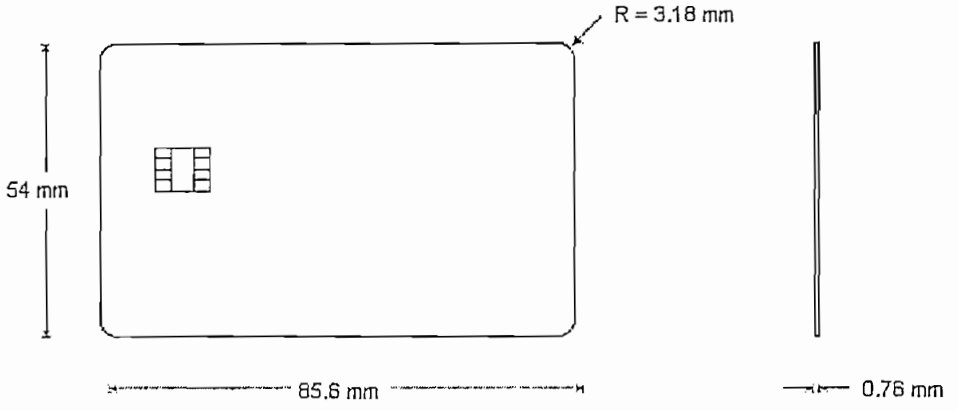

Figura 2.2 El formato ID-1: Espesor: 0,76 mm ± 0,08 mm; radio de la esquina: 3,18 mm ± 0,30 mm. Las dimensiones mostradas indican el lado de la tarjeta excluyendo tolerancias. (*Referencia:* [WOLFGANG])

El formato ID-000 es también definido usando dos rectángulos concéntricos. Toda vez que este formato se originó en Europa (basado en el sistema de telefonía móvil GSM), las dimensiones básicas se dan en unidades métricas. La esquina inferior derecha de una tarjeta plug-in es cortada con un ángulo de 45°, como se muestra en la Figura 2.3 para facilitar la correcta inserción de la tarjeta en el lector.

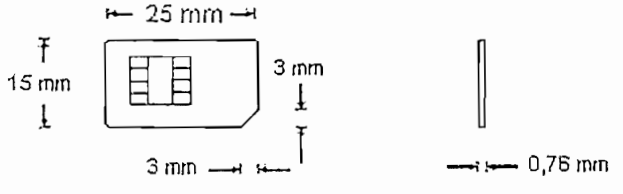

Figura 2.3 Formato ID-000. (*Referencia:* [WOLFGANG])

Las dimensiones de los dos rectángulos para el formato ID-000 son:

- Rectángulo externo largo: 25,10 mm
  ancho: 15,10 mm
- Rectángulo interno largo: 24,90 mm
  ancho: 14,90 mm

El formato ID-00 se basa también en unidades métricas. Sus dimensiones máximas y mínimas son definidas por dos rectángulos concéntricos con las siguientes dimensiones:

- Rectángulo externo largo: 66,10 mm
  ancho: 33,10 mm
- Rectángulo interno largo: 65,90 mm
  ancho: 32,90 mm

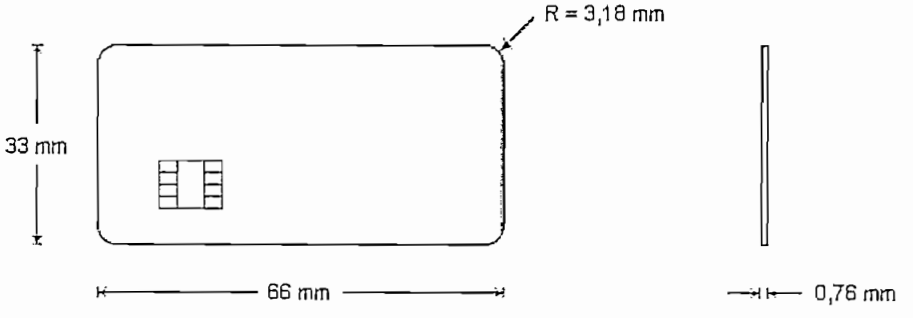

Figura 2.4 Formato ID-00. (*Referencia:* [WOLFGANG])

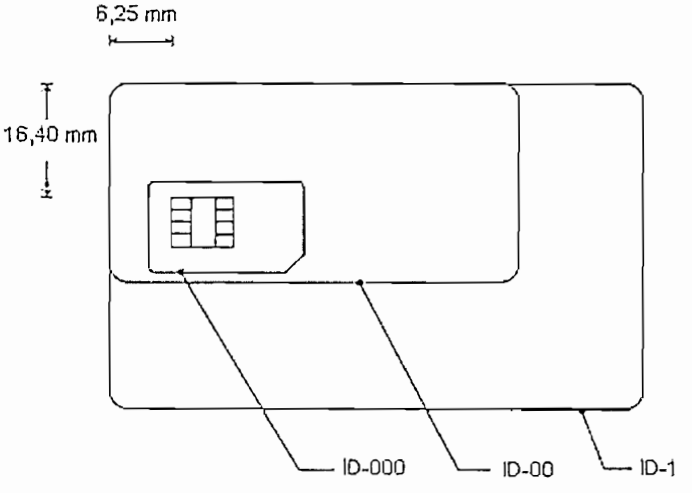

Figura 2.5 Tamaño relativo de los formatos ID-1, ID-00 e ID-000. (*Referencia:* [WOLFGANG])

- Resistencia eléctrica (de los contactos).- Todas las resistencias medidas entre dos puntos cualquiera de los pines no debe ser mayor que 0,5 Ohm, con cualquier valor de corriente que esté entre 50 μA y 300 mA,

- Interferencia electromagnética (entre la banda magnética y el circuito integrado),

- Campo electromagnético.- En esta prueba, la tarjeta es movida en un campo electromagnético estático con una fuerza de 1000 Oe (79.6 H) con una velocidad máxima de 1 cm/s. El chip de la tarjeta no debe sufrir daño,

- Electricidad estática.- La tarjeta no debe sufrir daño por una descarga eléctrica de 1500 V de un capacitor de 100 pF a través de una resistencia de 1500 Ohm. La prueba de esta propiedad se muestra en la Figura 2.8. La especificación requiere que este voltaje se descargue a través de cada uno de los contactos en polaridad normal e inversa. La TI debería aun ser operacional al final de la prueba.

La ISO está trabajando en función de mejorar esta parte del estándar ya que es muy importante. Adicionalmente, las tres pruebas mas usadas aplicadas por los fabricantes están especificadas en un anexo del estándar y son:

- Propiedades de encorvamiento
- Propiedades de torsión,
- Electricidad estática,

Ésta es una forma de comparar tarjetas fabricadas por diferentes companías, para ver si soportan cualquier relación de uso de una TI.

Las propiedades de encorvamiento se prueban doblando la tarjeta en cada eje como se muestra en la Figura 2.6, con una periodicidad de 30 dobleces por

minuto la tarjeta es doblada a 2 cm del centro de su eje largo y 1 cm de su eje pequeño. La prueba recomendada requiere que la tarjeta resista 250 dobleces en cada una de las cuatro orientaciones posibles. (son 1000 curvaturas en total).

Las propiedades de torsión de la tarjeta son probadas desplazando la tarjeta ± 15° respecto del eje largo con una periodicidad de 30 torsiones por minuto (Figura 2.7). El estándar requiere que la tarjeta soporte 1000 torsiones sin que el chip en la tarjeta falle o ella muestre un agrietamiento visible.

Uno de los puntos también referente a la manipulación de la TI guarda relación con el rango de temperatura durante su uso operacional. El estándar ISO/IEC 7810 define que la tarjeta ID-1 debería ser estructuralmente confiable y utilizable en el rango de - 35°C a + 50°C.

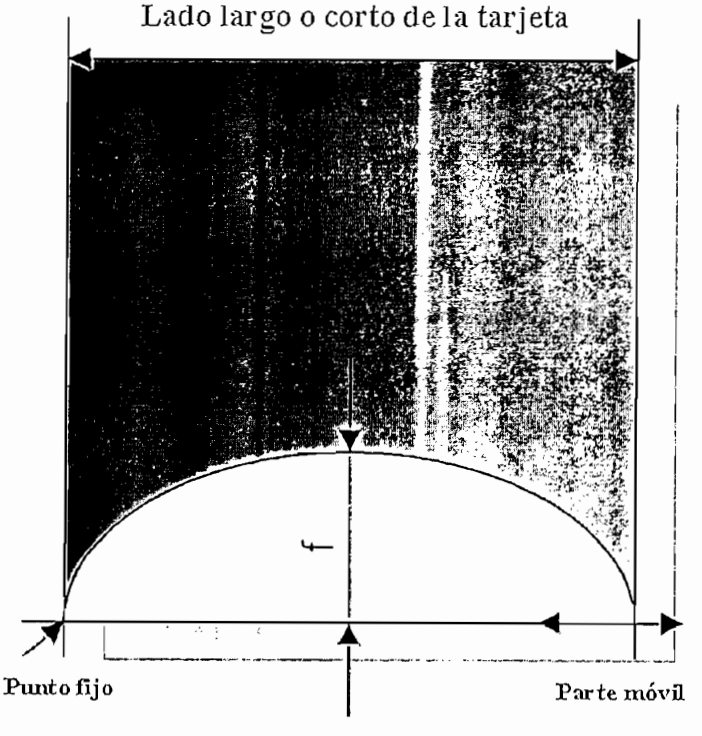

Figura 2.6 Prueba de encorvamiento. (*Referencia:* [FREETUTORIALS])

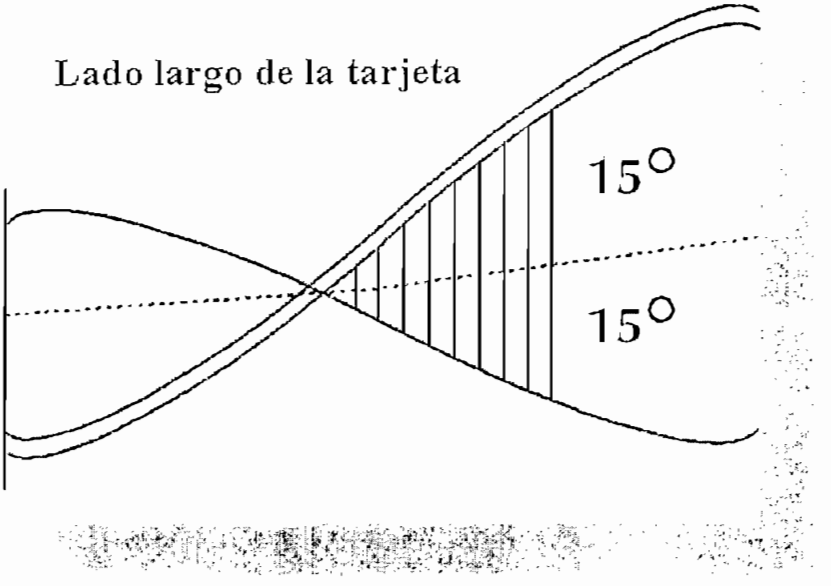

Figura 2.7 Prueba de torsión. (*Referencia:* [FREETUTORIALS])

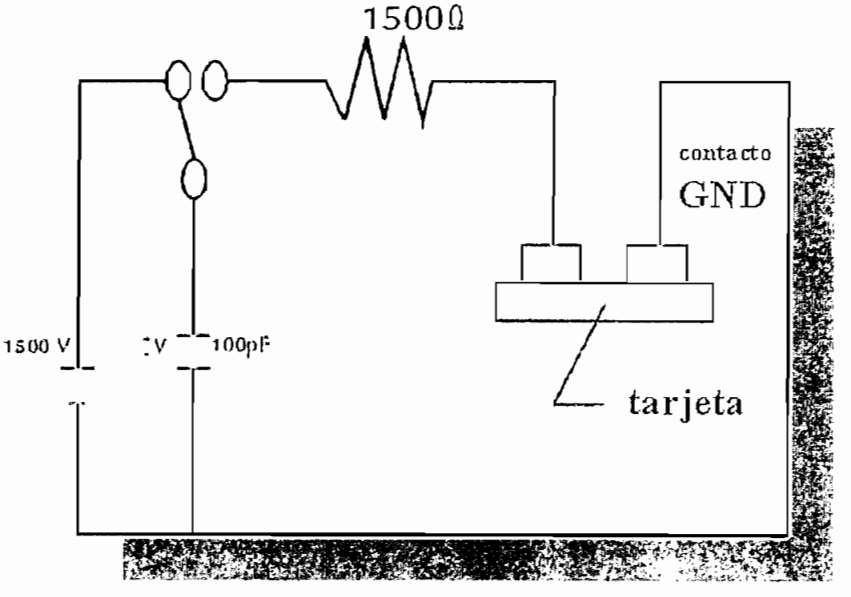

Figura 2.8 Prueba de resistencia a una descarga eléctrica estática. (*Referencia:* [FREETUTORIALS])

2.1.3 LOCALIZACIÓN DE LOS CONTACTOS

La principal diferencia entre una TI y otros tipos de tarjetas es el microcontrolador incrustado. Si los contactos se usan para el suministro de energía y la transmisión de datos, se requiere por tanto que existan conexiones eléctricas entre el chip incrustado y los contactos.

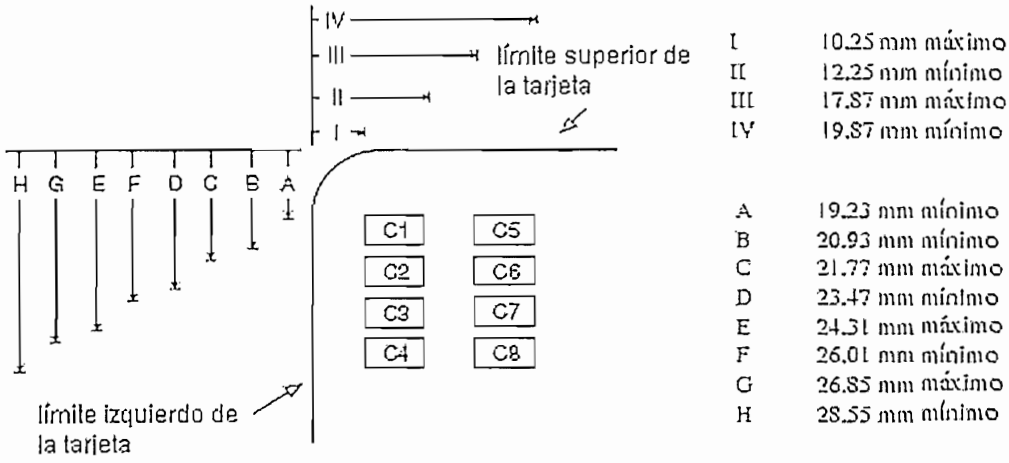

Figura 2.9 Posición de los contactos relativa al cuerpo de la tarjeta. (*Referencia:* [WOLFGANG])

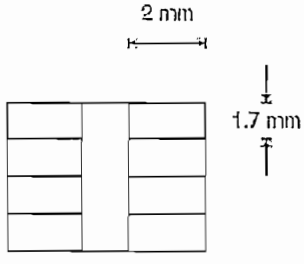

Figura 2.10 Dimensiones mínimas de los contactos como se especifica en ISO/IEC 7816-2. (*Referencia:* [WOLFGANG])

Aquellos como se ha dicho antes son seis u ocho contactos dorado-plateados, los cuales pueden ser vistos sobre cada TI estándar. La localización de esos contactos con respecto al cuerpo de la tarjeta y sus tamaños, están especificados en el estándar ISO 7816-2.

La posición absoluta del campo de contactos es la esquina superior izquierda del cuerpo de la tarjeta. La Figura 2.9 muestra un dibujo con las dimensiones especificadas. Las dimensiones mínimas de cualquier contacto son 1,7 mm por 2.mm (alto por ancho) Figura 2.10. Las dimensiones máximas de cualquier contacto no están especificadas, pero por supuesto están limitadas por el hecho de que los contactos individuales deben estar eléctricamente aislados unos de otros.

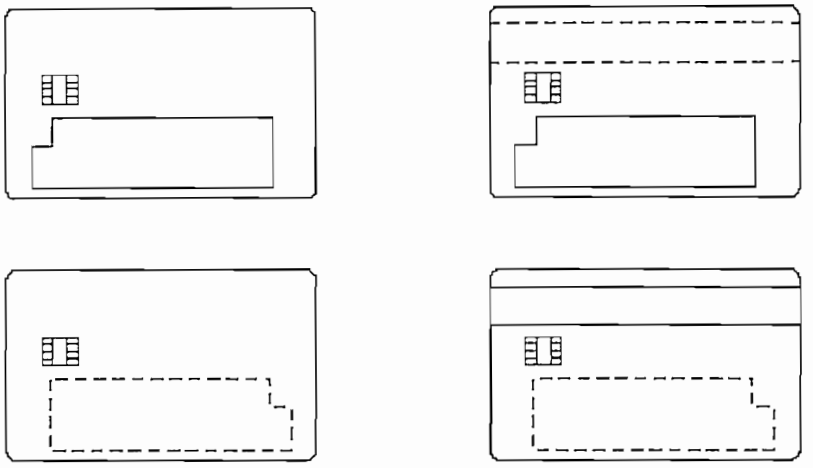

Figura 2.11 Diferentes formas de acomodar el chip, el campo de estampado en relieve y la banda magnética en una TI, de acuerdo con el estándar ISO 7816-2. (*Referencia:* [WOLFGANG])

La posición del módulo en el cuerpo de la tarjeta está especificada en el estándar. La localización del área de la banda magnética y el área reservada para el campo de estampado en relieve también están exactamente especificadas. Estos tres componentes podrían estar presentes en una misma tarjeta. Sin embargo, en este caso la siguiente relación mutua entre estos componentes debe ser tomada en cuenta: (a) si solo un chip y un campo para estampado en relieve están presentes, ellos podrían estar localizados en el mismo lado o en lados opuestos de la tarjeta; (b) si una banda magnética también está presente, ésta y el área para el estampado en relieve deben ser localizadas en lados opuestos de la tarjeta.

2.1.4 PROCESO DE FABRICACIÓN DE TIs

La fabricación de tarjetas inteligentes abarca un número diverso de pasos, que se describen a continuación:

1. *Fabricación del chip o muchos chips en forma de una pastilla u hostia.* Varios miles de circuitos integrados se fabrican a la vez en la forma de pastillas u hostias de silicio. Un chip individual para una TI es aproximadamente de 25 milímetros cuadrados o cerca de 5 milímetros por lado (cada vez estos valores

son menores)[2]. La plantilla para la circuitería en un chip se repite muchos millares de veces sobre una hostia de silicio de aproximadamente 4 pulgadas de diámetro. Tal hostia puede contener rutinariamente 3.000 a 4.000 chips cuando termina. La fabricación real de los chips en la hostia de silicio se hace con un proceso altamente refinado de deposición de vacío del material extremadamente puro del semiconductor en el substrato del silicio.

2. *Empaquetado de los chips individuales para la inserción en una tarjeta.* Una vez que se termina una hostia, cada chip individual en la misma se debe probar para cerciorarse de que es operable. Cada buen chip es identificado por una marca física en la preparación para cortar la oblea en muchos millares de pedazos (es decir, un chip por pedazo). Una vez que los chips son separados, un conector eléctrico que es más grande que el chip se adjunta. Los conectores eléctricos muy minúsculos (alambres) ligan las varias áreas en este conector a los pines específicos en el chip. La configuración que resulta se denomina *módulo*. La Figura 2.13 ilustra los componentes de un módulo, incluyendo las conexiones microelectrónicas entre el conector y el chip.

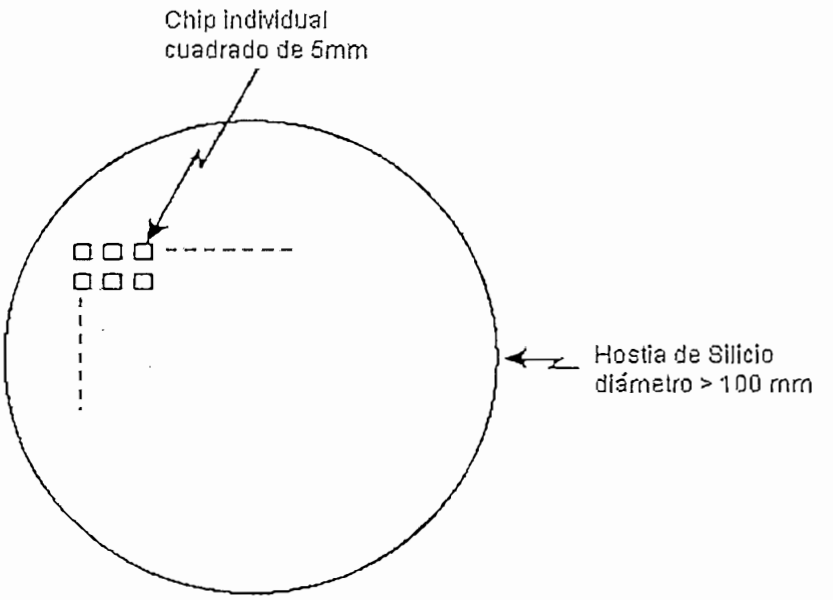

Figura 2.12 Fabricación del chip. (*Referencia:* [GUTHERY])

[2] *Referencia:* [WOLFGANG]

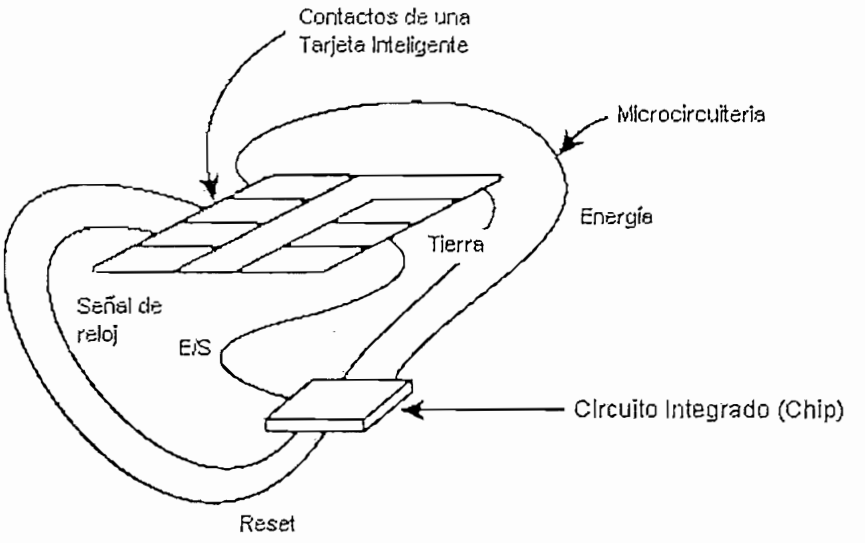

Figura 2.13 Elementos de un módulo de Tarjeta Inteligente. (*Referencia:* [GUTHERY])

3. *Fabricación de la tarjeta.* La tarjeta como tal se construye con cloruro de polivinilo o de un material similar como se verá más tarde. Las características químicas y las dimensiones de la tarjeta y de sus tolerancias asociadas son reguladas en su totalidad por los estándares internacionales (citados anteriormente). El material de la tarjeta se produce en una hoja grande, plana, del grueso prescrito. Muchos tipos de tarjetas son producidas en serie, luego estas hojas se imprimen. Las tarjetas individuales luego se separan de esta hoja plana y los bordes de cada tarjeta se alisan.

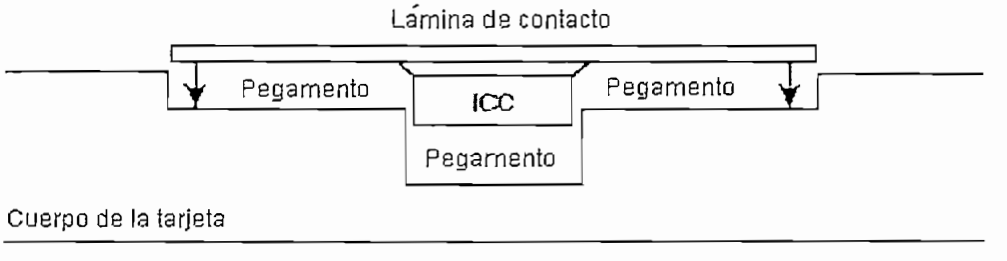

Figura 2.14 Inserción del módulo en el cuerpo de la tarjeta (*Referencia:* [GUTHERY])

4. *Inserción del chip en la tarjeta.* Una vez que el módulo y la tarjeta estén preparados, los dos se reúnen durante una operación de inserción. Un agujero se hace en la tarjeta y el módulo se pega en él. Este agujero es producido

típicamente con una operación de pulido o derritiendo el material y empujando el módulo directamente en él, como se muestra en la Figura 2.14 se inserta el módulo.

5. *Pre-personalización*. Una vez que el módulo se inserte en la tarjeta, la mayoría de las aplicaciones de la TI requieren que ciertos programas o archivos de datos estén instalados en cada chip (tarjeta) antes de que la tarjeta se pueda personalizar y dar a un titular específico de la misma. Esta preparación general del software o de los archivos en la tarjeta se hace con una operación llamada *pre-personalización*, que se hace a través de los conectores de E/S en la superficie de la tarjeta y por lo tanto puede proceder solamente a la velocidad apoyada por ese interfaz.

6. *Personalización*. El procedimiento de la personalización implica el poner la información tal como nombres y números de cuenta en el chip de la tarjeta. Esto también exige generalmente escribir un número de identificación personal (PIN) en la tarjeta que el titular puede utilizar para confirmar su identidad a la tarjeta. El procedimiento de personalización implica generalmente también la manipulación física de la tarjeta; es decir, fotos, nombres, dirección se imprimen a menudo en la tarjeta. Además, cierta información, tal como números de cuenta, se puede imprimir en alto relieve en la tarjeta para permitir la transferencia física de esa información sobre otros medios (por ejemplo, un recibo de papel después de una transacción de una tarjeta de crédito).

7. *Impresión de la tarjeta*. La impresión de gráficos y de texto en una TI es una característica extremadamente importante. El aspecto de la tarjeta refleja generalmente estéticamente y financieramente al emisor de la tarjeta. La información que se imprime tal como símbolos e insignias corporativos construye el reconocimiento para el emisor y tiene valor de publicidad significativo. Cuando una tarjeta se utiliza como tarjeta de identificación personal, la fotografía de una persona junto con la información del nombre y de la dirección se imprime a menudo en la tarjeta. Para muchas tarjetas que apoyan transacciones financieras,

los emisores se defienden a menudo de la amenaza de falsificación de sus tarjetas; y para esto hacen uso tanto como sea necesario de mecanismos anti-falsificación tales como hologramas impresos en la cara de la tarjeta.

Dependiendo de la información que se colocará en una tarjeta, una variedad de procesos de impresión se puede utilizar. Para las tarjetas que tendrán exactamente los mismos gráficos en todas, tal como tarjetas del teléfono, fichas de tránsito y similares, el paso de impresión se hace a menudo antes de la inserción del circuito integrado en la tarjeta. En este caso, las tarjetas son formadas desde láminas plásticas grandes. La impresión se hace en la lámina, antes de cortar las tarjetas individuales de la misma. Después de la operación de impresión, las tarjetas individuales son estampadas también en la lámina y sus bordes se alisan para coincidir con las especificaciones de la ISO.

8. *Inicialización del programa y de la información del programa sobre el chip en la tarjeta.* Éste viene a ser el paso final antes de que la tarjeta sea entregada a su titular.

Los fabricantes generalmente producen tarjetas en blanco en solo un formato (preferiblemente ID-1), con los módulos incrustados en ella y totalmente personalizada. Dependiendo de la aplicación específica para la que va a servir la tarjeta, la misma podría ser desprendida para el formato deseado en un paso de producción subsiguiente. Alternativamente, el formato de la tarjeta podría ser modificado por el cliente mas tarde, esto ha llegado a ser una práctica común con las tarjetas para teléfonos móviles. El cliente recibe una tarjeta ID-1 que está preperforada tal que ésta puede ser convertida en una tarjeta ID-000, para conseguir una pequeña tarjeta libre del cuerpo de una más grande. En otra técnica, la tarjeta ID-000 es perforada completamente y liberada de la tarjeta ID-1, luego se adjunta a ella solo con una cinta adhesiva la cual puede ser desprendida luego por el cliente, de esta forma el fabricante solo habrá producido un único formato de tarjeta como fuente de los otros.

2.1.5 MATERIALES DE LA TARJETA

El primer material empleado para las tarjetas ID, el cual aun es ampliamente usado es el cloruro de polivinilo (PVC) un material amorfo termoplástico. Éste es el menos caro de todos los materiales disponibles, fácil de procesar y adaptar para un amplio rango de aplicaciones, es usado en todo el mundo en tarjetas de crédito[3]. Sus inconvenientes son un tiempo de vida limitado, debido a su deterioro físico y resistencia limitada al frío y al calor. El PVC se usa en forma de láminas para fabricar tarjetas, puesto que el moldeado de inyección no es posible. La producción mundial de PVC fue alrededor de 13 millones de toneladas métricas en 1996, de las cuales 35.000 toneladas métricas (0,27%) fueron usadas en tarjetas. El PVC es considerado ambientalmente peligroso, dado que su materia base, el cloruro de vinilo es un conocido agente carcinógeno. Adicionalmente, si éste es quemado emana ácido clorhídrico y posiblemente dioxinas. Además, a menudo son usados compuestos de metales pesados como estabilizadores. El PVC es aun y por mucho el material para tarjetas más ampliamente usado. Esto, principalmente por su bajo costo y buenas características de procesamiento. Sin embargo, cada año es menor el uso de PVC debido a sus indeseables propiedades ambientales[4]. Muchos usuarios de tarjetas han decidido no usar tarjetas hechas con PVC por razones de políticas ambientales.

Para evitar los inconvenientes del PVC, el ABS (acrylonitrile butadiene styrene) ha sido usado en ocasiones para hacer tarjetas. Se trata también de un termoplástico amorfo que se distingue por su estabilidad y resistencia a temperaturas extremas. Consecuentemente se usa muy a menudo en la fabricación de tarjetas para teléfonos móviles, los cuales por obvias razones son expuestos a relativamente altas temperaturas. El ABS puede ser procesado tanto en forma de lámina como por moldeado de inyección. Sus mayores inconvenientes son su limitada aceptación a la tinta y su baja resistencia al desgaste. Aunque el material básico para la producción de ABS, benceno, es un

[3] *Referencia:* [WOLFGANG]
[4] *Referencia:* [WOLFGANG]

agente carcinógeno, el ABS no tiene ningún otro inconveniente ambiental conocido.

Para aplicaciones en las cuales se requiere estabilidad y durabilidad extremas, se usa el Poli carbonato (PC). Este material es típicamente usado para tarjetas de identificación y este dicho sea de paso es el material base para la fabricación de discos compactos y DVDs. Esto es debido a su alta estabilidad térmica, se requieren temperaturas relativamente altas para la aplicación de hologramas o bandas magnéticas usando el proceso de estampado al calor. Los principales inconvenientes del poli carbonato son su bajo grado de resistencia a ser raspado y muy alto costo comparado con otros materiales que se usan para fabricar tarjetas. Un inconveniente adicional es que el fosgeno y el cloro son necesarios para la producción de poli carbonato y ambos materiales son problemáticos para el medio ambiente. El poli carbonato puede ser fácilmente reconocido por el sonido metálico característico que produce cuando cae sobre una superficie dura.

Un material amigable con el medio ambiente principalmente usado para sustituir el PVC es el PET (polyethylene terephthalate), el cual ha sido usado por un tiempo relativamente largo para hacer materiales de empaquetado. Éste es comúnmente conocido como poliéster. Este material termoplástico es usado en TIs tanto en su forma amorfa (A-PET) como en su forma cristalina (PETP). Ambos tipos son adecuados para el procesamiento, en forma de láminas e igual de bien por moldeado de inyección. Sin embargo, el PETP es difícil de laminar, lo cual hace que sean necesarios pasos adicionales de procesamiento en el proceso de fabricación.

Numerosos esfuerzos se han realizado buscando nuevos o mejores materiales para el cuerpo de la tarjeta junto a los materiales usuales (PVC, ABS, PC y PET). Un ejemplo es acetato celuloso, el cual aunque tiene buenas propiedades ambientales, hasta ahora ha probado ser pobremente adecuado para la producción en masa de tarjetas. Realmente diferentes materiales, tales como papel, han sido frecuentemente discutidos, pero no han sido usados en

cantidades significativas. Los requerimientos impuestos sobre las tarjetas, en términos de costo, durabilidad y calidad, son después de todo muy altos y ellos pueden satisfacerse en la actualidad solo con lo que se conoce como plásticos.

2.2 PROPIEDADES ELÉCTRICAS

Las propiedades eléctricas de una TI dependen solamente del microcontrolador incrustado en ella. Dado que éste es el único componente de la tarjeta con un circuito eléctrico. Indudablemente esta situación cambiará en el futuro con la adición de nuevos componentes a la tarjeta, tales como displays, teclados, etc. Pero pasará algún tiempo antes de que esas tarjetas sean ampliamente usadas.

La aplicación que desde su inicio ha impuesto muchos requerimientos y muy rígidos sobre las propiedades eléctricas de las TIs es el sistema GSM que se usa en telecomunicaciones móviles. Este sistema, el cual caracteriza una extremadamente larga variedad de tipos técnicamente diferentes de dispositivos terminales hechos por igualmente una gran variedad de fabricantes, los cuales deben trabajar con una variedad de tipos de tarjetas, a impuesto por mucho tiempo requerimientos extremadamente severos. Debido al gran número de TIs usadas en el sistema GSM, las características eléctricas especificadas para tarjetas GSM han llegado a ser un lineamiento general para todos los fabricantes de microcontroladores para TIs. Por esta razón se puede asumir que casi todos los microcontroladores nuevos cumplirían con los parámetros eléctricos generales de las especificaciones relevantes GSM, dado que de otro modo las TIs serían inutilizables en el mercado de las telecomunicaciones.

En los primeros días de la tecnología de la TI, muy a menudo la primera consideración fue que el microcontrolador incrustado fuese multifuncional y menor atención fue puesta a sus características eléctricas generales, tales como consumo de corriente. En aquel entonces, las aplicaciones fueron casi exclusivamente cerradas y usaban un solo tipo de tarjeta con un terminal especialmente diseñado para ajustarse a aquel tipo de tarjeta. Las propiedades

Consecuentemente, los requerimientos para el menor consumo de corriente posible y bajo suministro de voltaje son altamente importantes en esta área de aplicación.

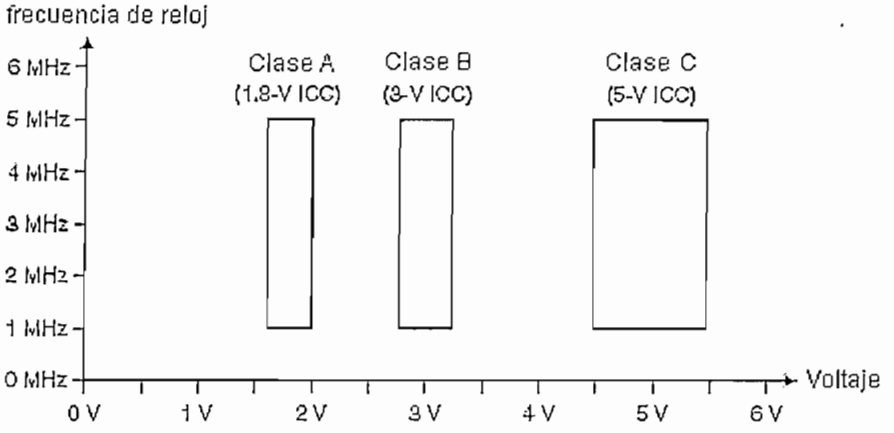

Figura 2.15 Comparación de las 3 clases de voltaje y rangos de frecuencia fijados como se especifica en el estándar ISO/IEC 7816-3 y en la enmienda 1. (*Referencia:* [WOLFGANG])

La Tabla 2.1 provee un resumen de los requerimientos eléctricos más importantes de los estándares internacionales e industriales esenciales.

2.2.1 CONEXIONES ELÉCTRICAS

Las **TIs** tienen ya sea seis u ocho contactos sobre el lado frontal, los cuales forman el interfaz eléctrico entre el terminal y el microcontrolador en la tarjeta. Todas las señales eléctricas se pasan a través de esos contactos. Sin embargo, de acuerdo a ISO/IEC 7816-2, dos de los ocho contactos (C4 y C8) están reservados para los contactos auxiliares AUX1 y AUX2, los cuales podrían ser usados en el futuro para interfaces tales como USB.

El componente más importante de una TI naturalmente es el chip. Claro está que este frágil componente no puede ser simplemente laminado a la superficie de la tarjeta como una banda magnética. En lugar de eso, necesita una especie de cerco para protegerlo del maltrato diario durante su tiempo de vida. Este cerco es conocido como módulo del chip. Adicional a la protección de las condiciones del ambiente, los chips para tarjetas con contactos necesitan seis u ocho contactos como se dijo antes. Una porción de la superficie del módulo sirve

para proveer esos contactos eléctricos al mundo exterior. Naturalmente, el módulo del chip debería ser entre otras cosas tan barato como fuera posible.

Tabla 2.1 Resumen de los tres parámetros eléctricos (voltaje, corriente y tasa de reloj) para los estándares internacionales e industriales más importantes. (*Referencia:* [WOLFGANG])

Estándares y clases	Voltaje	Tasa de reloj	Corriente máxima
ISO/IEC 7816-3 y enmienda 1			
Clase A	5 V ± 10% => 4,5 – 5,5 V	1-5 MHz	60 mA a 5 MHz
Clase B	3 V ± 10% => 2,7 – 3,3 V	1-5 MHz	50 mA a 4 MHz
Clase C	1,8 V ± 10% => 1,62 – 1,98 V	1-5 MHz	30 mA a 4 MHz
Clase A, B y C con el reloj parado.			0,5 mA (reloj en stop)
EMV 2000	5 V ± 10% => 4,5 – 5,5 V	1-5 MHz	50 mA para todas las tasas de reloj
GSM 11.11 5-V SIM	5 V ± 10% => 4,5 – 5,5 V	1-5 MHz	10 mA (en operación) 200 µA (en estado idle) 200 µA (reloj en stop)
GSM 11.12 3-V SIM	3 V ± 10% => 2,7 – 3,3 V	1-5 MHz	6 mA a 3,3V / 5 MHz (en operación) 200 µA a 1 MHz (en idle) 100 µA (reloj en stop)
GSM 11.18 1,8-V SIM	1,8V ± 10% => 1,62 – 1,98 V	1-5 MHz	4 mA a 1,8 V / 5 MHz (en operación) 200 µA a 1 MHz (en idle) 100 µA (reloj en stop)
TS 102.221			
Clase A desde el reset hasta la selección de la aplicación	5 V ± 10% => 4,5 – 5,5 V	1-5 MHz	10 mA a 5 MHz (en operación) 200 µA a 1MHz (en idle)
Clase A durante una sesión de una aplicación específica.			60 mA a 5 MHz
Clase B desde el reset hasta la selección de la aplicación	3 V ± 10% => 2,7 – 3,3 V	1-5 MHz	7,5 mA a 5 MHz o 6 mA a 4 MHz (en operación) 200 µA a 1 MHz (en idle)
Clase B durante una sesión de una aplicación específica.			50 mA a 5 MHz
Clase C desde el reset hasta la selección de la aplicación	1,8 V ± 10% => 1,62 – 1,98 V	1-5 MHz	5 mA a 5 MHz (en operación) 4 mA a 4 MHz (en operación) 200 µA a 1 MHz (en idle)
Clase C durante una sesión de una aplicación específica.			30 mA a 5 MHz

En el presente, algunos módulos de **TIs** tienen solo seis contactos, esto reduce ligeramente los costos de producción. Sin embargo, éstos tienen las mismas propiedades que los módulos de ocho contactos. Los contactos son numerados secuencialmente desde el tope izquierdo hasta el fondo derecho. La Figura 1.2 muestra la designación ISO y la asignación eléctrica de los ocho contactos definidos.

Hasta finales de los ochentas, fue necesario aplicar un voltaje externo para programar y borrar la EEPROM, dado que los controladores usados entonces no tenían bombas de carga. El contacto C6 fue reservado para este propósito. Sin embargo, desde inicios de los 90s éstas han sido prácticamente estandarizadas para generar este voltaje directamente en el chip usando una bomba de carga, por tanto este contacto no ha sido usado desde hace mucho. No obstante, éste no puede ser empleado por algunas otras funciones, ya que éstas podrían tener conflictos con el estándar ISO. De este modo, toda TI tiene un contacto que no tiene una función real, pero el cual debe mantenerse presente. Toda vez que el contacto de voltaje de programación yace entre otros dos que son necesarios para la operación de la tarjeta, éste no puede ser simplemente eliminado. Esto reduce en algo los inconvenientes de tener un contacto superfluo.

Como es lógico, se requieren conexiones eléctricas entre el chip dentro del módulo y los contactos en la cara externa del módulo. Actualmente, se usan dos procesos principales para este fin. En el proceso wire-bonding, una máquina automática para juntar los cables suelda cables de oro con un diámetro de solo unos pocos micrómetros al chip y a la parte posterior de los contactos. Los cables se juntan eléctricamente al chip y al módulo usando soldadura ultrasónica. Con este proceso, el contacto se acomoda en la superficie superior del chip siempre en forma opuesta al módulo. Éste ha sido un proceso estándar en la industria de los semiconductores por algún tiempo y puede ser fácilmente usado para la producción en masa de módulos de chip. Sin embargo, cada chip debe ser conectado eléctricamente al módulo usando cinco cables, lo cual naturalmente cuesta tiempo y dinero.

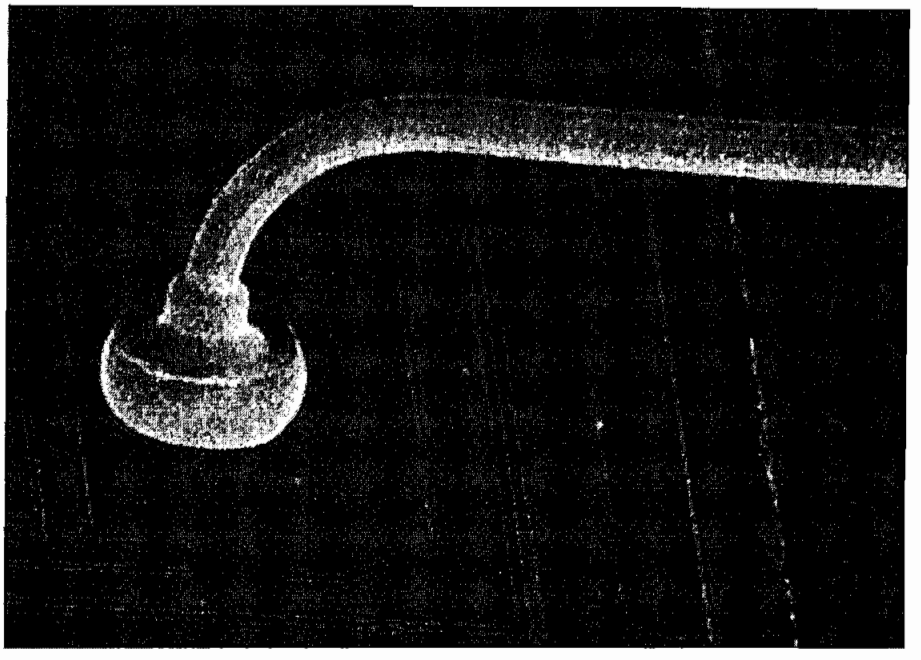

Figura 2.16 Fotografía de la zona de contacto entre un cable y una almohadilla del microcontrolador de una TI, ampliada 1000 veces (*Referencia:* [WOLFGANG])

El proceso die-bonding fue desarrollado para fomentar la reducción del costo de la adaptación del chip en el módulo. En este proceso, las conexiones eléctricas entre el chip y el módulo no se hacen con cables. En lugar de eso, las conexiones se hacen por la unión mecánica del chip a la superficie posterior del módulo.

2.2.2 FUENTE DE VOLTAJE

El suministro de voltaje para una **TI** en un inicio fue de 5 V, con una tolerancia máxima de ± 10 %. Este voltaje, el cual es el mismo usado convencionalmente para los circuitos TTL, fue el valor estándar para todas las **TIs** y todas las aplicaciones. Como con otros componentes semiconductores, su estructura progresivamente se va haciendo más pequeña y por consiguiente se necesita reducir el consumo de corriente lo que hace necesaria una marcada reducción del rango de voltaje. En este sentido se han hecho grandes esfuerzos por parte del sector de la telefonía móvil. La demanda del mercado obliga una reducción del peso de los teléfonos móviles requiriéndose por tanto para su alimentación cambiar baterías de 6V por las de 3V. Dado que todos los

componentes de teléfonos móviles estuvieron disponibles en la tecnología de 3V, por un momento las **TIs** fueron el único componente en telefonía móvil que aun necesitaba 5V. Consecuentemente, un costoso convertidor de voltaje fue necesario para proveer energía eléctrica a la **TI**, resultando en un costo extra que podía ser evitado.

Consecuentemente, en los estándares internacionales el rango de voltaje de las **TIs** fue primero extendido a 3 – 5V con una tolerancia de ± 10 %. Esto da como resultado un rango efectivo de 2,7 a 5,5 V. Sin embargo, es de esperarse que esta extensión sea insuficiente. Consecuentemente, la versión revisada de ISO/IEC 7816-3 y la enmienda 1 ISO/IEC 7816-3 otra vez serán revisadas en un futuro cercano para permitir a las **TIs** recibir un suministro de voltaje de 1,8 V con una tolerancia de ± 10 %.

La ampliación del rango de voltaje no supone un problema para los microprocesadores ni para la mayoría de tipos de memorias, particularmente porque el voltaje básico para los semiconductores construidos con la tecnología de 0,13 µm es usualmente solo 1,8 V. Sin embargo, las EEPROMs son también integradas dentro del microcontrolador de la **TI**. Estas EEPROMs y sus bombas de carga asociadas forman el más grande obstáculo para las **TIs** de bajo voltaje. A pesar de todo, con cierta cantidad de ingenio técnico es ciertamente posible integrar EEPROMs y sus bombas de carga dentro de microcontroladores que puedan trabajar con un suministro de voltaje en el rango de 1,62 V a 5,5 V.

El estándar ISO/IEC 7816-3 y su enmienda establecen tres clases para caracterizar los rangos de voltaje de las **TIs**. La Clase A cubre el rango de voltaje de 5V ± 10 %. La Clase B cubre el rango de 3V ± 10 % y la Clase C cubre el rango de 1,8V ± 10%. Estas tres clases pueden ser utilizadas individualmente o ser combinadas según se desee. Por ejemplo, si una **TI** conoce los requerimientos tanto de la Clase A como de la Clase B, ésta puede ser utilizada tanto con suministro de voltaje de 5V como de 3V. Sin embargo, ésta debe tener en mente que el rango entre 3,3V y 4,5V yace fuera del rango especificado, por lo tanto no es necesario que se permita a la tarjeta trabajar en este rango. Por otro

deseada. Si la ATR no incluye ninguna información acerca del rango de voltaje, la TI podría ser usada con el voltaje con el cual la primera ATR pudo ser recibida. Es un hecho que una clase adicional de voltaje será introducida en el futuro para habilitar una fuente de voltaje de 1,2V

2.2.3 SUMINISTRO DE CORRIENTE

El microcontrolador de la tarjeta obtiene su suministro de voltaje y de esta manera su fuente de corriente, por medio del contacto C1. De acuerdo a la especificación GSM, esta corriente no debería exceder los 10 mA. La primera versión del estándar ISO/IEC 7816-3 en 1989 especificó una corriente máxima de 200 mA con una fuente de voltaje de 5V y un reloj de 5 MHz. Pero incluso en ese entonces aquello fue demasiado. Desde ese momento, los valores han sido significativamente reducidos y de hecho dependen de las varias clases de voltajes.

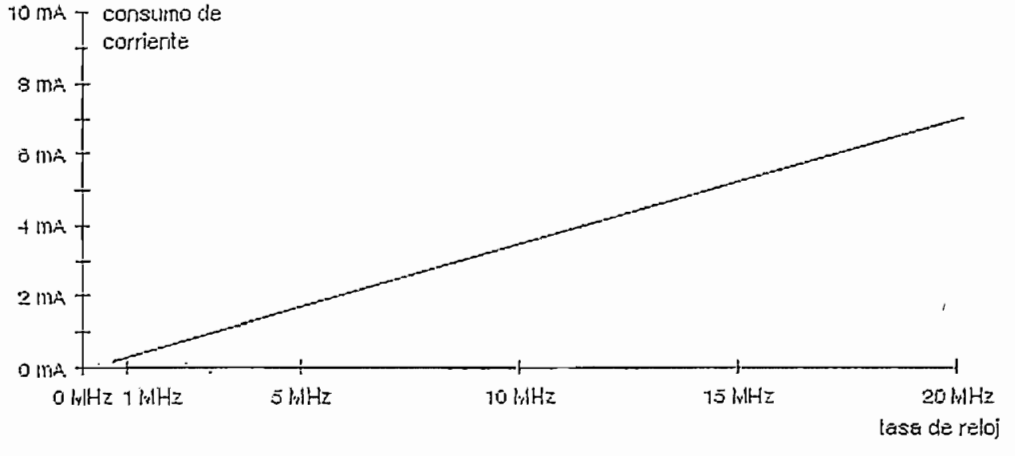

Figura 2.17 Consumo de corriente en el microcontrolador versus frecuencia de reloj en el modo de operación normal. El consumo de corriente en el modo de dormido con el reloj aplicado también es linealmente dependiente de la frecuencia de reloj y es aproximadamente 50 μA a 5 MHz, dependiendo del tipo de microcontrolador. (*Referencia:* [WOLFGANG])

El factor importante es que el consumo de corriente es directamente proporcional a la frecuencia de reloj aplicada y al suministro de voltaje, éste también de alguna forma depende de la temperatura del microcontrolador. La versión actual de la especificación ISO/IEC 7816-3 especifica una corriente máxima de 60 mA para un voltaje Clase A (5V) una frecuencia de reloj máxima de 5 MHz y una temperatura ambiente máxima de 50 °C. Con respecto a las TIs para

transacciones financieras, en la especificación EMV 2000 el valor ISO/IEC 7816-3 para la corriente máxima se reduce de 60 mA a 50 mA, pero no existen otras restricciones suplementarias significantes. En el sector de telecomunicaciones, el consumo de corriente ha sido un factor crítico desde el principio. Consecuentemente en este sector existe un complicado conjunto de reglas especificando la corriente máxima como una función de la tasa de transferencia (clock rate) y estados de operación de la TI.

Una innovación técnicamente interesante ha sido introducida para TIs que conforman las especificaciones USIM. Aquí dos diferentes estados de operación son especificados con respecto al consumo de corriente. El primer estado abarca el tiempo desde el reset hasta la selección de la aplicación. Mientras el segundo estado comprende la subsiguiente sesión de la aplicación específica. La máxima corriente permisible en el primer estado es significativamente más baja que en el segundo estado. Además el teléfono móvil puede usar un objeto de datos (data object) de una aplicación específica para determinar la demanda de corriente de la aplicación seleccionada en ese momento. Esto es porque el consumo de corriente de la TI es considerablemente más alto cuando su coprocesador numérico o el multiplicador interno de frecuencia están habilitados para activar un alto desempeño. Con estos mecanismos, podría al menos teóricamente ser posible tener un teléfono móvil que use solo estas "aplicaciones inteligentes" en las cuales el consumo de corriente puede ser adecuadamente soportado. Desafortunadamente la especificación USIM no incluye ningún procedimiento para permitir que un teléfono móvil y una TI negocien la máxima corriente disponible, como con PPS[7]. Con la versión actual de la especificación, la única opción disponible para un teléfono móvil si su TI demanda mucha corriente es desactivarla.

Los microcontroladores modernos para TIs tienen un consumo de corriente en el orden de 350μA por MHz de frecuencia de reloj[8]. Usando este valor, podemos escribir la siguiente fórmula para el consumo de corriente del

[7] [PPS] Parámetro de selección de protocolo
[8] *Referencia:* [WOLFGANG]

microcontrolador en función de la frecuencia de reloj aplicada o la frecuencia de reloj generada dentro del chip:

$$I = \frac{f}{2.875} \bullet \frac{mA}{MHz}$$

Esta fórmula es útil para calcular un valor estimado inicial, pero hay que recordar que el consumo de corriente no solo depende de la frecuencia del reloj, sino también del suministro de voltaje, la temperatura y por supuesto del tipo de chip.

Con un suministro de voltaje de 5V y asumiendo un consumo de corriente de 60mA, una TI tiene un consumo de potencia de 300 mW. Este valor es tan bajo que no existe necesidad de tomar en cuenta un sobrecalentamiento del chip mientras está en operación. Aunque esta cantidad de potencia es disipada sobre un área de aproximadamente 20 mm^2.

Todos los microcontroladores de TIs tienen uno o más modos de ahorro de energía. El principio de operación de dichos modos se basa en la desactivación de todos los componentes funcionales del chip que no están siendo usados. En principio, solo la interrupción lógica del interfaz E/S los registros del procesador y la RAM necesitan mantenerse energizadas para guardar el estado de configuración que está siendo usado. En la práctica, el procesador a menudo también se mantiene energizado, pero la ROM y la EEPROM están apagadas. Cuando el procesador está en el modo de dormido o en estado idle, el consumo en ese momento cae dramáticamente, dado que la mayoría de partes del chip están aisladas de la fuente de voltaje.

Además para el modo sleep, muchos microcontroladores de TIs soportan otro modo en el cual el reloj aplicado puede estar apagado, este modo se conoce como "clock stop mode". El propósito principal de este modo es habilitar a los componentes de hardware en el terminal para que generen la señal de reloj que está apagada, lo cual hace de este modo particularmente atractivo para modo de operación del dispositivo terminal.

De acuerdo al estándar ISO/IEC 7816-3, la corriente máxima permisible en el modo de dormido con el reloj en stop es 500μA para las tres clases. Aunque este valor es muy alto para el área de telecomunicaciones móviles.

Otro detalle importante en relación con el suministro de corriente que causa severos dolores de cabeza a los fabricantes de terminales que escogen ignorarlo, es que todos los microcontroladores actualmente emplean tecnología CMOS. Bajo ciertas consideraciones, largos corto circuitos pueden ocurrir brevemente durante procesos de encendido y apagado del transistor. Eso produce picos en ese momento que son muchas veces más grandes que la corriente de operación nominal, con una duración en el rango de los nanosegundos. Esos picos pueden también ocurrir cuando la bomba de carga de la EEPROM se enciende. Si el terminal no puede suministrar corrientes tan altas durante esos cortos intervalos la fuente de voltaje caería bajo el valor permitido. Esto puede producir un error de escritura en la EEPROM o disparar el detector de bajo voltaje en el chip. Por esta razón, referencias de tales picos pueden ahora ser encontrados en prácticamente cada estándar y especificación relevantes.

2.2.4 RELOJ EXTERNO

Los procesadores de las TIs no tienen generadores internos de reloj. Por tanto es necesaria una señal de reloj externo. Este reloj también provee la referencia para la tasa de transferencia de datos. De acuerdo al estándar ISO/IEC 7816-3 y la mayoría de otros estándares y especificaciones, el factor de trabajo del reloj debe ser 50%. La tolerancia usual es un rango de factor de trabajo de 40 a 60%.

La señal de reloj aplicada al contacto no es necesariamente la misma que el reloj interno provee al procesador. Algunos microcontroladores tienen un circuito multiplicador o divisor que puede opcionalmente ser insertado entre el reloj interno y externo. El divisor generalmente tiene un factor de división de 2, lo que causa que la tasa del reloj interno sea solo la mitad de la tasa del reloj externo. Esto es en parte debido a las características del hardware del chip y en

parte porque permite a los osciladores ya presentes en los terminales ser usados como la fuente de la señal de reloj para el chip.

La mayoría de microcontroladores permite que la señal de reloj sea apagada cuando el CPU esté en el modo de dormido. En este caso, apagar el reloj significa mantener la línea de reloj en un nivel definido. Dependiendo de lo que el fabricante del chip prefiera, el nivel de "apagado" podría ser bajo o alto.

Toda vez que las **TIs** succionan solo unos pocos microamperios desde la línea de reloj, apagar el reloj podría a primera vista parecer un poco curioso. Sin embargo, la cantidad de potencia ahorrada dentro del terminal es substancial, por tanto esto puede ser meritorio en ciertas aplicaciones.

2.2.5 DESCRIPCIÓN DE LOS CONTACTOS

En esta sección se muestran algunos valores de parámetros eléctricos aplicados a los distintos contactos acorde lo especificado en el estándar ISO/IEC 7816-3.

Las siguiente es una lista de abreviaciones que se utilizarán en esta sección:

Vih : Voltaje de entrada nivel alto (High level input voltage)
Vil : Voltaje de entrada nivel bajo (Low level input voltage)
Vcc : Suministro de voltaje (Power supply voltage at VCC)
Vpp : Voltaje de programación (Programming voltage at VPP)
Voh : Voltaje de salida nivel alto (High level output voltage)
Vol : Voltaje de salida nivel bajo (Low level output voltage)
tr : Tiempo de elevación de la amplitud de la señal del 10% al 90 % (Rise time).
tf : Tiempo de caída de la amplitud de la señal del 90% y 10 % (Fall time).
Iih : Corriente de entrada nivel alto (High level input current)

Iil : Corriente de entrada nivel bajo (Low level input current)
Icc : Suministro de corriente a VCC (Supply current at VCC)
Ipp : Corriente de programación a VPP (Programming current)
Ioh : Corriente de salida nivel alto (High level output current)
Iol : Corriente de salida nivel bajo (Low level output current)
Cin : Capacitancia a la entrada (Input capacitance)
Cout : Capacitancia a la salida (Output capacitance)

2.2.5.1 Contacto E/S

Como se ha dicho antes este contacto se usa como entrada (modo de recepción) o salida (modo de transmisión) para el intercambio de datos. Dos posibles estados existen para el contacto E/S.

- Marca o estado alto (estado Z), si la tarjeta y el terminal están en modo de recepción o si este estado es impuesto por el transmisor.

- Espacio o estado bajo (estado A), si este estado es impuesto por el transmisor.

Cuando los dos extremos de la línea están en modo de recepción, la línea debería permanecer en estado Z. Cuando los dos extremos están en modos de transmisión desiguales, el estado lógico de la línea podría ser indeterminado. Durante las operaciones, el terminal y la tarjeta no pueden estar en modo de transmisión a la vez.

2.2.5.2 Contacto Vpp

Este contacto se usa para suministrar el voltaje requerido para programar o para borrar la memoria interna no volátil. Dos posibles estados existen para VPP: El estado idle y el estado activo, como se define en la Tabla 2.3. El estado idle podría ser mantenido por el terminal sin que el estado activo sea requerido.

Tabla 2.2 Características eléctricas de E/S bajo condiciones de operación normales. (*Referencia:* [CARDWERK])

SÍMBOLO	CONDICIONES		MÍNIMO	MÁXIMO	UNIDADES
Vih	Ambos (1)	Iih max = ± 500 µA	2	VCC	V
	O	Iih max = ± 50 µA	0,7 VCC	VCC (3)	V
Vil		Iil max = ± 1 mA	0	0,8	V
Voh (2)	Ambos	Iih max = ± 100 µA	2,4	VCC	V
	O	Iol max = ± 20 µA	3,8	VCC	V
Vol		Iol max = ± 1 mA	0	0,4	V
Tr, tf	Cin = 30 pf ; Cout = 30 pf			1	µs

(1) Para el terminal se toma en cuenta ambas condiciones.
(2) Se asume que un resistor se usa en el dispositivo interfaz (20 Ohm es el valor recomendado)
(3) El voltaje sobre E/S debe mantenerse entre 0.3 V y VCC + 0,3 V

Tabla 2.3 Características eléctricas de VPP bajo condiciones de operación normales. (*Referencia:* [CARDWERK])

SÍMBOLO	CONDICIONES	MÍNIMO	MÁXIMO	UNIDADES
Vpp	Estado IDLE	0,95*VCC	1,05*VCC	V
Ipp	(programación no activa)		20	mA
Vpp	Estado activo	0,975*P	1,025*P	V
Ipp	(programación de la tarjeta)		I	mA
Los valores por defecto son: P=5 e I=50				

2.2.5.3 Contacto CLK

La frecuencia entregada por el dispositivo terminal sobre la línea CLK, está diseñada ya sea por la frecuencia inicial fi durante la ATR o por la frecuencia subsiguiente fs durante la transmisión subsiguiente. El ciclo de trabajo para operaciones asincrónicas debería estar entre 40% y 60% del período de operación estable. Se debería tener cuidado cuando se cambia de frecuencia (de fi a fs) para asegurarse de que ningún pulso es más corto que el 45% del período mas corto.

Tabla 2.4 Características eléctricas de CLK bajo condiciones de operación normales. (*Referencia:* [CARDWERK])

SÍMBOLO	CONDICIONES		MÍNIMO	MÁXIMO	UNIDADES
Vih	Ambos (1)	Iih max = ± 200 µA	2,4	VCC (2)	V
	O (1)	Iih max = ± 20 µA	0,7*VCC	VCC (2)	V
	O	Iih max = ± 10 µA	VCC-0,7	VCC (2)	V
Vil		Iil max = ± 200 µA	0 (2)	0,5	V
tr, tf	Cin = 30 pf			9% del período con un max de 0,5 µs	

(1) Para el terminal toma en cuenta las tres condiciones.
(2) El voltaje sobre E/S debe mantenerse entre 0.3 V y VCC + 0,3 V

2.2.5.4 Contacto RST

Tabla 2.5 Características eléctricas de RST bajo condiciones de operación normales. (*Referencia:* [CARDWERK])

SÍMBOLO	CONDICIONES		MÍNIMO	MÁXIMO	UNIDADES
Vih	Ambos (1)	Iih max = ± 200 µA	4	VCC (2)	V
	O	Iih max = ± 10 µA	VCC-0,7	VCC (2)	V
Vil		Iil max = ± 200 µA	0 (2)	0,6	V

(1) Para el dispositivo interfaz toma en cuenta ambas condiciones.
(3) El voltaje sobre E/S debe mantenerse entre 0.3 V y VCC + 0,3 V

2.2.5.5 Contacto VCC

Este contacto se usa para proporcionar el voltaje de poder VCC, el cual como se dijo antes puede ser Clase A, Clase B o Clase C,

Tabla 2.6 Características eléctricas de VCC (Clase A) bajo condiciones de operación normales. (*Referencia:* [CARDWERK])

SÍMBOLO	MÍNIMO	MÁXIMO	UNIDAD
VCC	4,75	5,25	V
ICC		200	mA

2.2.6 PROCEDIMIENTO PARA LA TRANSMISIÓN DE DATOS

Comencemos analizando lo siguiente, si un error ocurre durante la transmisión de datos, puede ocurrir que el terminal y la tarjeta intenten enviar datos al mismo tiempo. El resultado es una colisión de datos en la línea E/S. Muy a parte del problema en el nivel de aplicación, en el nivel físico podrían producirse corrientes en la línea que podrían ser lo suficientemente grandes para destruir el interfaz. Para prevenir el daño del semiconductor en tales eventos, la línea de E/S en el terminal está ligada al nivel de +5V vía una resistencia de 20kΩ, como se muestra en la Figura 2.18, esto, en combinación con la convención acordada de no enviar nunca un nivel activo 5V evita cualquier problema que podría ocurrir si las dos partes intentan manejar la línea de datos a dos diferentes niveles como resultado de un error de comunicación. Cuando sea que la línea de E/S tenga que ser puesta en un nivel +5V durante las comunicaciones, la parte en cuestión simplemente pone su salida en un estado de alta impedancia y la línea se levanta al nivel de +5V por la resistencia.

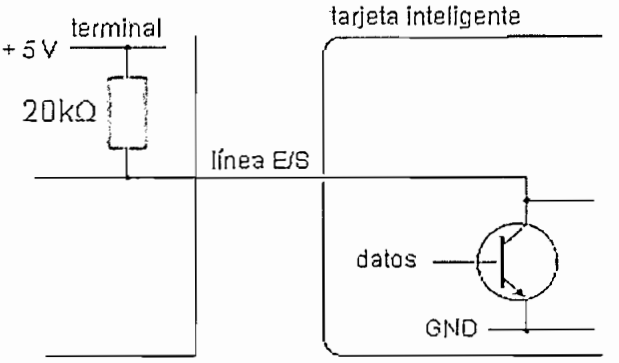

Figura 2.18 Circuito del canal E/S entre el terminal y la tarjeta inteligente. (*Referencia:* [WOLFGANG])

2.2.6.1 Secuencias de activación y desactivación

Todos los microcontroladores de TIs están protegidos contra cargas electrostáticas sobre los contactos. Precisamente para evitar estados indefinidos, secuencias específicas de activación y desactivación están predefinidas, a las que se debe estar estrictamente apegado. Esto también es reflejado en la parte relevante del estándar ISO/IEC 7816-3. Estas secuencias definen los aspectos eléctricos de activación y desactivación de la tarjeta y no tienen nada que ver con

la secuencia de establecimiento de contacto mecánico con la tarjeta, lo cual de todas maneras no está especificado. No obstante, el contacto mecánico se hace primero con el contacto de tierra de la tarjeta como una precaución inteligente para asegurar una conexión y desconexión eléctrica bien definida.

Como se muestra en la Figura 2.19 la conexión eléctrica de tierra debe ser la primera en hacerse, seguida por la conexión de la fuente de voltaje. Después de éstas viene la conexión de reloj. Si un intento de conectar el reloj se hizo antes de la conexión de la fuente de voltaje, por ejemplo, el microcontrolador podría tratar de succionar su suministro de corriente enteramente vía la línea de reloj. Esto podría irremediablemente dañar el chip, causando una falla funcional completa. Una secuencia de desactivación defectuosa podría también tener efectos similares en el microcontrolador.

Cuando el microcontrolador está en operación, éste puede reiniciar vía la línea de reset. Esto requiere que un nivel bajo se aplique en primer lugar a esta línea, con el reset siendo iniciado por la subsiguiente ligera subida a un nivel alto. Un reset durante la operación es conocido como un reset caliente igual que en los sistemas de computación.

Por otro lado, un reset frío ocurre cuando todas las líneas de suministro se colocan en los estados que especifica el estándar ISO. La operación de reseteo de la tarjeta se detalla más adelante.

2.2.6.1.1 Conexión y activación de los contactos

Los circuitos eléctricos no deben ser activados hasta que los contactos estén conectados al terminal esto como se dijo antes para evitar posibles daños a cualquier tarjeta que quiera establecer una sesión.

La activación de los contactos debe consistir de las siguientes operaciones consecutivas:

- RST está en estado L;
- VCC debe ser energizado;
- E/S en el terminal debe ser colocado en el modo de recepción;
- VPP será elevado a su condición de idle;
- CLK será provisto de una señal de reloj confiable y estable.

2.2.6.1.2 Reset de la tarjeta

Un reset de la tarjeta se inicia por el terminal, después de lo cual la tarjeta responderá con una ATR. Al finalizar la activación de los contactos (RST está en L, VCC energizado y estable, E/S en modo de recepción en el terminal, VPP estable en nivel idle, CLK provisto de una señal de reloj estable y confiable), la tarjeta contesta asincrónicamente que está lista para el reset.

La señal de reloj se aplica al contacto CLK al tiempo T0. La línea E/S será puesta en estado Z dentro de 200 ciclos de reloj de la señal de reloj (t2) que está siendo aplicada al contacto CLK (tiempo t2 luego de T0).

Un reset interno resetea la tarjeta después de unos pocos ciclos de reloj. La ATR sobre la línea E/S debe llegar entre 400 y 40000 ciclos de reloj (t1) después de que la señal de reloj ha sido aplicada al contacto CLK (el tiempo t1 después de T0).

Una tarjeta con un reset bajo en ejecución es reseteada manteniendo a RST en estado L por al menos 40000 ciclos de reloj (t3) después de que la señal de reloj ha sido aplicada al contacto CLK (el tiempo t3 después de T0). De este modo si ninguna ATR arriba dentro de 40000 ciclos de reloj (t3) con RST en estado L, RST se pone en estado H (al tiempo T1). La ATR sobre E/S deberá arribar entre 400 y 40000 ciclos de reloj (t1) después de la señal sobre el contacto RST suba a su nivel alto (el tiempo t1 luego de T1). Si la ATR no arriba dentro de 40000 ciclos de reloj (t3) con RST en estado H (el tiempo t3 luego de T1), la señal sobre el contacto RST deberá ser devuelta al estado L (al tiempo T2) y el contacto deberá ser desactivado por el terminal.

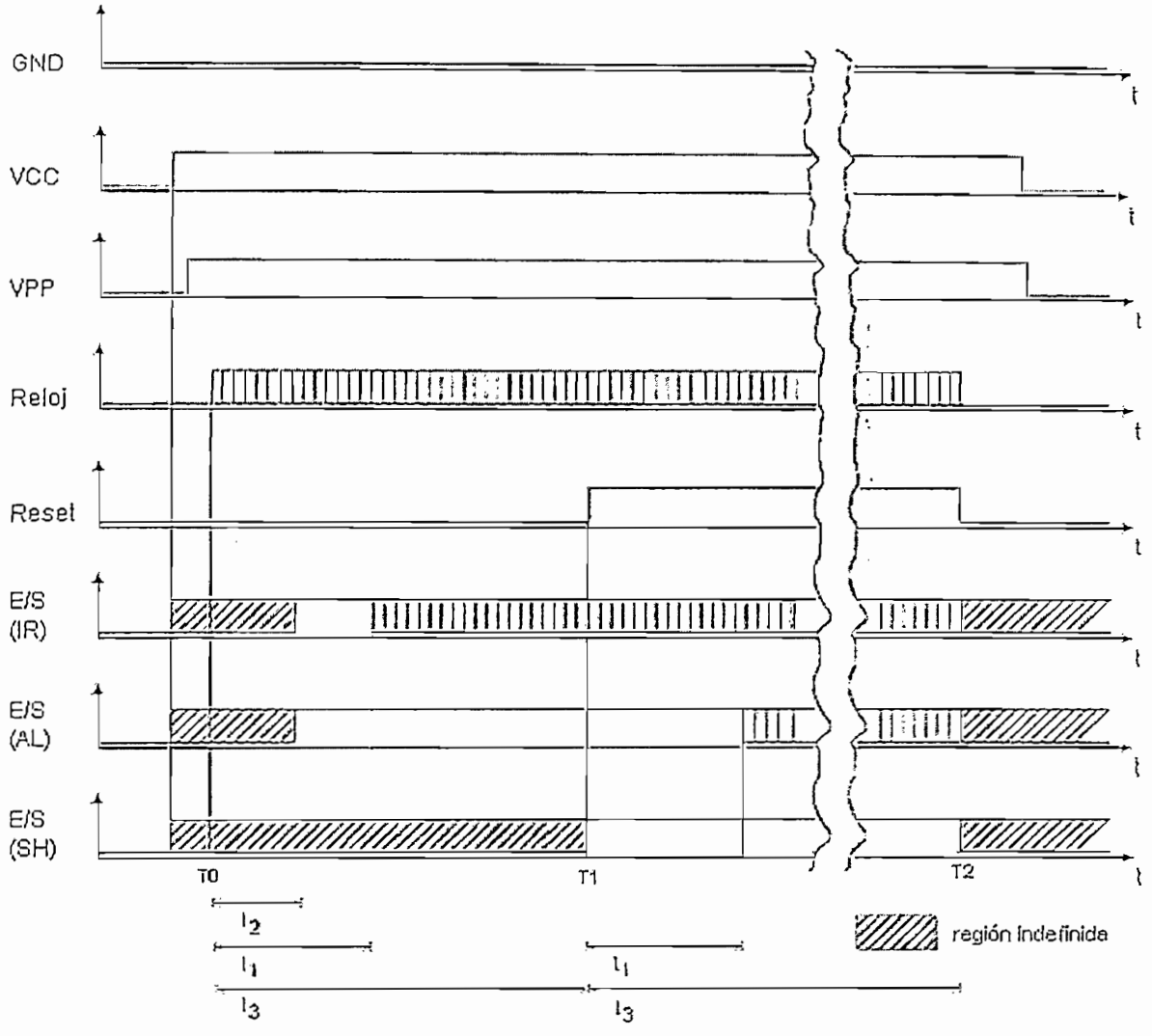

Figura 2.19 Secuencias de activación y desactivación de las tarjetas inteligentes acorde al estándar ISO/IEC 7816-3, los intervalos t1 y t2 yacen en los rangos 400/f $\leq$ t1 $\leq$ 40.000/f y t2 $\leq$ 200/f, 40.000/f $\leq$ t3. IR: Reset interno, AL: Reset asincrónico, SH: Reset sincrónico. (*Referencia:* [WOLFGANG])

Con una tarjeta que responde sincrónicamente, el terminal coloca todas las líneas a estado L (ver Figura 2.20). VCC es energizado, VPP es puesto en estado idle, CLK y RST se mantiene en estado L, E/S se pone en modo de recepción en el terminal, RST deberá mantenerse en estado H por al menos 50 µs (t12), antes del retorno otra vez al estado L.

El pulso de reloj es aplicado después de un intervalo (t10) desde la elevación al borde de la señal reset. La duración del estado H del pulso de reloj puede ser cualquier valor entre 10 µs y 50 µs; no se permite más que un pulso de

reloj mientras el reset está en alto. El intervalo de tiempo entre la bajada de los bordes sobre los contactos CLK y RST es t11.

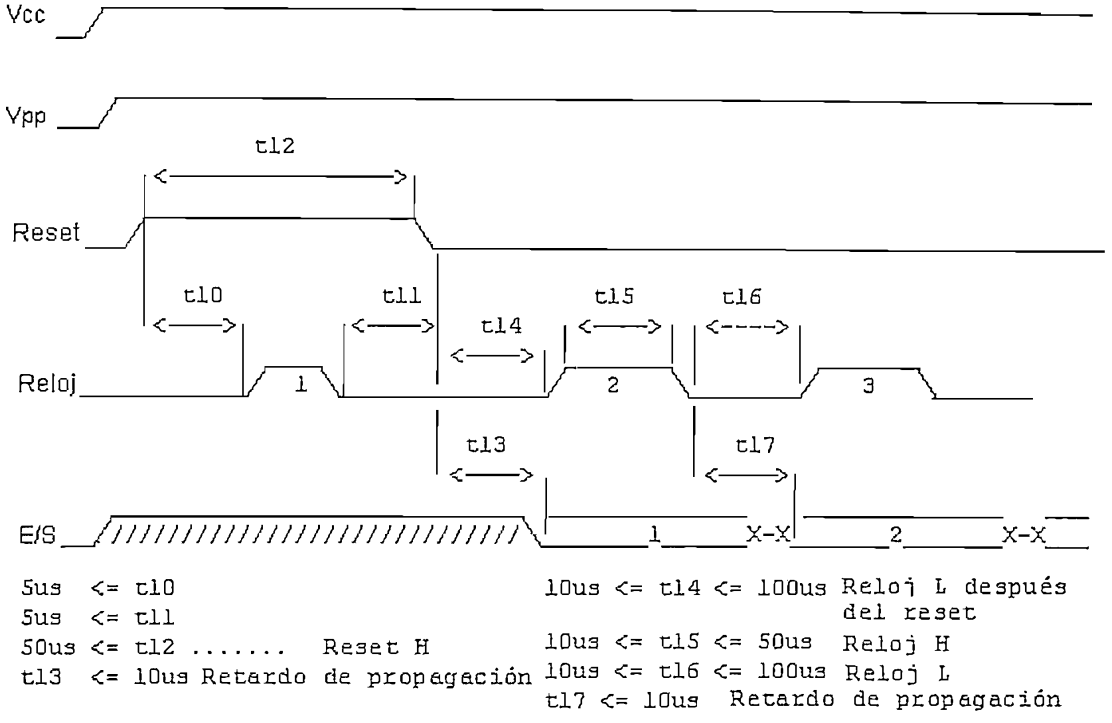

Figura 2.20 Reset de la tarjeta cuando espera una respuesta sincrónica. (*Referencia:* [CARDWERK])

El primer bit de datos se obtiene como una ATR sobre la línea E/S mientras CLK está en estado L y es válido después de un intervalo t13 desde la caída de la señal sobre RST.

NOTAS:

1) Se asume que el estado interno de la tarjeta está definido antes del reset. Por esto el diseño de la tarjeta tiene que evitar la operación impropia.

2) Un reset de la tarjeta puede ser iniciado por el terminal por su decisión en cualquier momento.

3) El terminal debe soportar uno o más de estos tipos de comportamiento de reset.

2.2.6.1.3 Desactivación de los contactos

Cuando se termina o aborta el intercambio de información (una tarjeta que no reacciona o se ha detectado que la tarjeta ha sido removida), los contactos eléctricos deberán ser desactivados.

La desactivación por el terminal consistirá de las siguientes operaciones consecutivas.

- Estado L sobre RST
- Estado L sobre CLK
- VPP inactivo
- Estado A sobre E/S
- VCC inactivo

2.3 MICROCONTROLADORES DE TIs

Desde la perspectiva informática, el componente central de una tarjeta inteligente es el microcontrolador incrustado bajo los contactos. Él controla, inicia y monitorea todas las actividades de la tarjeta. Los microcontroladores que han sido especialmente diseñados y desarrollados para este propósito son pequeños computadores. Esto significa que contienen procesadores, memoria e interfaces para el contacto con el mundo exterior.

Los componentes funcionales más importantes del típico microcontrolador de TIs son los procesadores, los buses de datos, los buses de dirección y los tres tipos de memoria (RAM, ROM y EEPROM). El chip también tiene una unidad interfaz que provee la comunicación serial con el mundo exterior. Esta interfaz no debería ser imaginada como una unidad funcional compleja que puede independientemente enviar y recibir datos. En el caso más simple, la interfaz serial es solo una localidad que puede ser direccionada por la CPU y está conectada al contacto E/S.

Las tecnologías de semiconductores que son en el presente mas comúnmente usadas para producir microcontroladores de TIs trabajan con estructuras de ancho aproximadamente de 0,25 μm, 0,18 μm y 0,13 μm, los cuales definitivamente caen en el rango de estructuras de ancho factibles actualmente más pequeñas[9].

Los microcontroladores usados en TIs no son componentes estándar, ampliamente disponibles. En lugar de eso, han sido desarrollados específicamente para este propósito y no se usan en otras aplicaciones. Existen algunas razones importantes para esto, las cuales se describen a continuación.

2.3.1 TIPOS DE PROCESADORES

Los procesadores usados en TIs no son diseños especiales, sino que en lugar de eso resulta que estos dispositivos han sido usados en otras áreas durante mucho tiempo. En esta industria no es usual desarrollar nuevos procesadores para áreas de aplicación especial, debido a que esto generalmente es muy costoso. Adicionalmente, los procesadores de TIs deben ser extremadamente confiables. Por lo tanto es mejor confiar en los tipos de procesadores más viejos que han sido probados en la práctica, en vez de experimentar con los últimos desarrollos de fabricantes de semiconductores. La industria aeroespacial, la cual está muy interesada en la seguridad funcional, usa solo componentes que están una o dos generaciones atrás del estado actual del desarrollo, por las mismas razones.

Los microcontroladores de TIs que están en lo mas bajo de la escala de desempeño usualmente tienen una memoria direccionable en el rango de 6 KB a 30 KB[10]. Bajo esas condiciones, el uso de un bus de memoria de 8 bits no impone ninguna restricción. El procesador utilizado generalmente tiene una arquitectura CISC (complex-instruction-set computer), lo cual significa que requieren algunos ciclos de reloj para ejecutar instrucciones de máquina y usualmente tienen sets de

[9] *Referencia:* [WOLFGANG]
[10] *Referencia:* [WOLFGANG]

instrucciones muy largos. El rango de direcciones de los procesadores 8 bits es a menudo 16 bits, lo cual permite que 65536 bytes puedan ser direccionados. Los sets de instrucciones del procesador están basados ya sea en la arquitectura de Motorola 6805 o Intel 8051. El fabricante de semiconductores podría añadir instrucciones suplementarias al set de instrucciones estándar. Tales instrucciones generalmente involucran opciones adicionales para direccionamiento de memoria con 16 bits, la cual existe solo en la forma más rudimentaria en los dos sets de instrucciones que forman la base para los sets de instrucciones de los procesadores en TIs.

La familia de procesadores de 8 bits está también disponible con extensiones que les permiten direccionar bancos de memoria adicional, para escalar sobre el límite de 64 KB. El acceso a tales bancos de memoria se controla mediante registros especiales que mapean los bancos de memoria dentro de una región de memoria específica, donde ella puede ser accedida por el procesador. Sin embargo, este tipo de direccionamiento de memoria no lineal tiene inconvenientes significantes. Por ejemplo, la distribución relativamente compleja del código de programa sobre algunos bancos de memoria problematiza significativamente el software, incrementando de esta forma la probabilidad de errores. También se requiere de memoria adicional para la funcionalidad de conmutación de bancos. Consecuentemente, la expansión de espacio de memoria usando bancos de memoria es principalmente un remedio temporal que se usa mientras se espera la transición a procesadores con mayor número de bits.

Los procesadores de 16 bits incluyen algunas derivaciones de las arquitecturas 8051 existentes, así como de diseños de companías específicas, tales como los dispositivos Renesas H8, Philips XA y Samsung CALM. Dicho sea de paso, por mucho tiempo el H8 fue el único procesador de 16 bits para microcontroladores de TIs con una arquitectura del tipo RISC y un correspondiente set de instrucciones ('RISC' representa 'reduced-instruction-set computer').

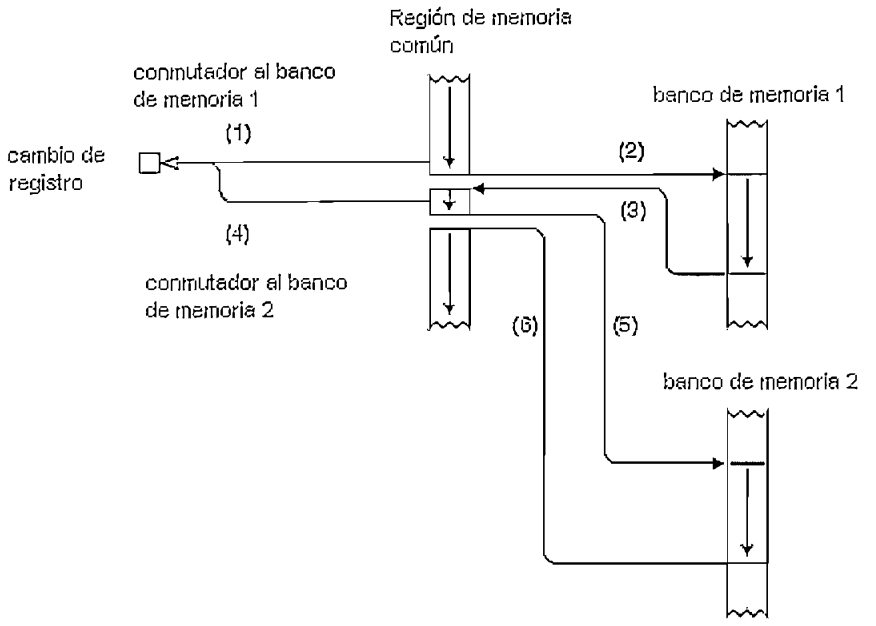

Figura 2.21 Acomodamiento básico de una memoria dividida en algunos bancos y el flujo de programa asociado. Este ejemplo muestra dos subrutinas localizadas en dos diferentes bancos de memoria que son llamadas desde una región común de memoria. Los números en paréntesis indican la secuencia de eventos en el proceso. (*Referencia:* [WOLFGANG])

En el extremo superior de la escala de desempeño para microcontroladores de TIs, tanto los tipos de 16 bits como los de 32 bits están disponibles en este momento. La tendencia del desarrollo muy claramente se dirige en dirección de los procesadores de 32 bits. Tales procesadores son necesarios urgentemente en este tipo de desempeño, para manipular grandes memorias (que exceden la frontera de los 64-kB) y sobre todo para satisfacer las enormes ansias de procesamiento de los COS modernos, tales como los de las Java Cards. El criterio de selección de llaves para los procesadores incluye densidad de código, disipación de energía y resistencia a ataques.

Entre los procesadores de 32 bits, los tipos de compañías específicas están llegando a ser establecidos en la actualidad, pero dos procesadores centrales usados en los mercados típicos del microcontrolador también han ganado un mercado en el reino de las TIs. Ellos son los procesadores MIPS y ARM.

El primer paso dentro del lenguaje de 32 bits tuvo lugar en 1993 con el proyecto European CASCADE ('Chip Architecture for Smart Card and portable

intelligent Devices')[11]. Uno de los objetivos de este proyecto fue el de proveer un procesador de alto desempeño para TIs. El proyecto seleccionó el procesador ARM 7M RISC, el cual a menudo se usa en dispositivos portátiles tales como cámaras de video. Tiene una arquitectura de 32 bits y puede correr hasta a 20MHz con un suministro de voltaje de 3V, mientras consume solo 40 mA. En la tecnología de 0,8 μm, el centro ARM 7 tiene un área de 5,9 mm² (3,12 x 1,9 mm), con el procesador aritmético asociado para los algoritmos usuales de llaves públicas (RSA, DSA y EC) ocupando un área adicional de 2 mm². El procesador tiene tanto el modo de supervisor como el usuario y por esto soporta particionamiento del código de COS y del código de aplicación. Desde el ARM 7 originalmente seleccionado, éste ha sido optimizado para aplicaciones de TIs y en la forma del SC 100 ('secure code') ha sido construido dentro de microcontroladores de TIs por varios fabricantes de semiconductores. El siguiente paso en la evolución es una variante del ARM 9 dedicado a TIs, con la designación de tipo SC 200. Una situación similar ocurre con los procesadores MIPS, los cuales también fueron desarrollados originalmente para otras áreas de aplicación. Aunque los procesadores de 32 bits tienen significativamente mas espacio muerto que los procesadores de 8 bits usando la misma tecnología, debido a sus anchos buses y sus estructuras internas más complejas, serán usados en un creciente número de aplicaciones con TIs en el futuro. El poder de procesamiento que ofrecen es indispensable para esas aplicaciones, por tanto la desventaja de mayor consumo de energía e incremento del área del chip son precios que se pueden pagar por el progreso. Por supuesto, los procesadores de 8 bits no desaparecerán en un futuro previsible, dado que proveen una sólida base para chips no muy caros en lo mas bajo de la escala de desempeño.

2.3.2 TIPOS DE MEMORIAS

Junto al procesador, los más importantes componentes de un microcontrolador son varios tipos de memoria, la cual sirve para almacenar código de programa y datos. Dado que los microcontroladores de las TIs deben ser

[11] *Referencia:* [WOLFGANG]

computadores completos, ellos exhiben una división característica de memoria en memoria RAM, ROM y EEPROM. La división exacta depende principalmente del área de aplicación del chip. En cualquier caso, siempre se hace un esfuerzo por mantener las memorias RAM y EEPROM tan pequeñas como sea posible, dado que ellas requieren el mayor espacio por celda.

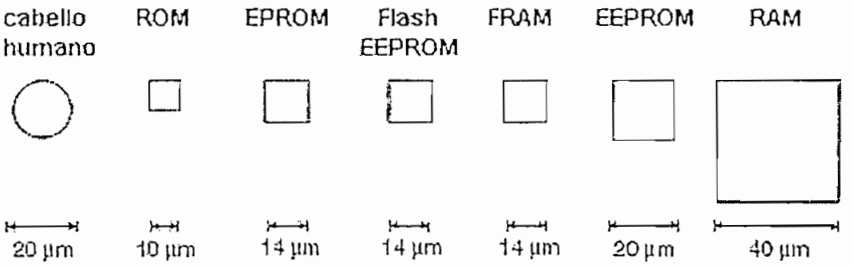

Figura 2.22 Comparación del área de una sola celda-bit para varios tipos de memoria. (*Referencia:* [WOLFGANG])

En el caso de TIs multiaplicación, las cuales administran algunas aplicaciones al mismo tiempo, los chips mas comúnmente usados tienen una capacidad de la ROM que es apenas dos veces mas grande que la EEPROM, para proveer espacio suficiente para almacenar el complejo código del SO. Para TIs de una sola aplicación, se seleccionan microcontroladores cuya capacidad de EEPROM es solo ligeramente superior al volumen de datos de la aplicación.

Todos los datos de variables de aplicación, junto con algunas partes del sistema operativo, pueden de este modo ser guardadas en la EEPROM para hacer un uso óptimo de la misma, la cual toma una cantidad relativamente grande de espacio sobre la matriz y es por lo tanto costosa.

Integrar tres tipos de memorias de semiconductor en una sola matriz de silicón es una tarea relativamente difícil que requiere de un número significante de pasos de producción y máscaras de exposición. Los diferentes tipos de memoria también ocupan áreas marcadamente diferentes, debido a sus estructuras y principios de operación diferentes.

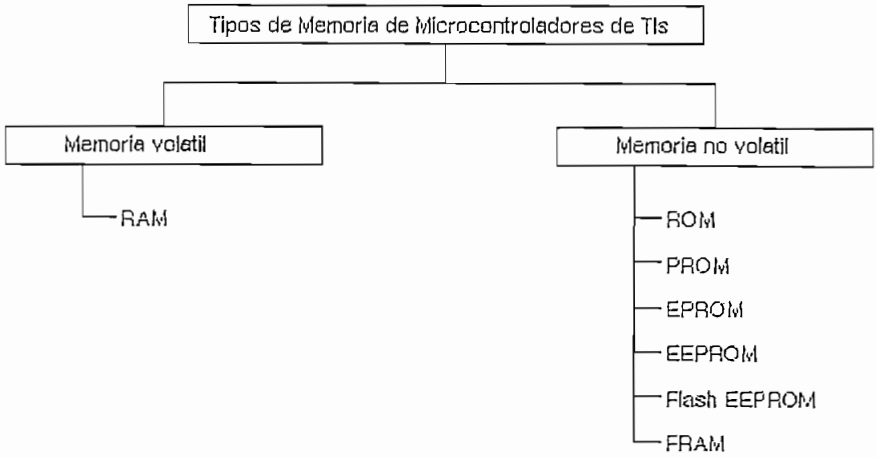

Figura 2.23 Clasificación de memorias de microcontroladores de TIs, las PROM y EPROM ya no se usan en microcontroladores modernos, las FRAM están empezando a usarse en TIs. (*Referencia:* [WOLFGANG])

Un nuevo tipo de tecnología de memoria para TIs ha llegado a estar disponible desde hace relativamente poco. Esta es llamada 'flash' EEPROM y permite tener tiempos de acceso para escritura y borrado que son mucho más cortos que con los tipos previos de memoria EEPROM. El tamaño de la celda es aproximadamente la mitad del tamaño de una celda de una EEPROM convencional (Figura 2.22), dependiendo de las particularidades del diseño.

2.3.2.1 Memoria ROM (read-only memory)

Como su nombre lo indica esta memoria solo puede ser leída mas no escrita, esto implica que no requiere energía para mantener los datos almacenados.

Una ROM de TIs contiene la mayoría de rutinas del COS, así como varias funciones de test y diagnóstico. Esos programas están construidos dentro del chip por su fabricante cuando éste es construido. Esto se hace por la preparación de una plantilla de la ROM del código de programa y luego usando esa plantilla para 'quemar' el programa en el chip usando un proceso litográfico. En este caso los datos, los cuales son los mismos para una carrera de producción, también pueden ser ingresados en la ROM en el proceso de fabricación.

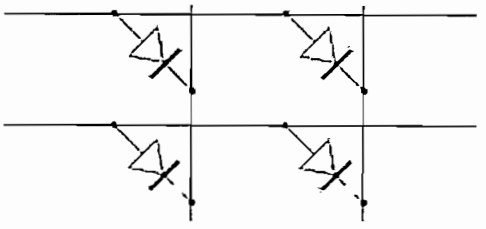

Figura 2.24 Estructura funcional básica de una ROM. (*Referencia:* [WOLFGANG])

2.3.2.2 Memoria EEPROM (electrically erasable programmable read-only memory)

La EEPROM la cual es técnicamente más compleja que la ROM o RAM, se usa en TIs para todos los datos y programas que necesitan ser modificados o borrados en algún momento. Funcionalmente, una EEPROM corresponde al disco duro de una PC, dado que éste retiene los datos en ausencia de energía y los datos pueden ser modificados siempre que se necesite. La EEPROM por lo tanto es una memoria no volátil.

En principio, una celda EEPROM es un diminuto capacitor que puede ser cargado o descargado. El estado de carga puede ser interrogado por un sensor de estados lógicos. Un capacitor cargado representa un 1 lógico, mientras que un capacitor descargado representa un 0 lógico. Para almacenar un byte de datos, son necesarios ocho de estos pequeños capacitores, junto con la adecuada circuitería para el sensor.

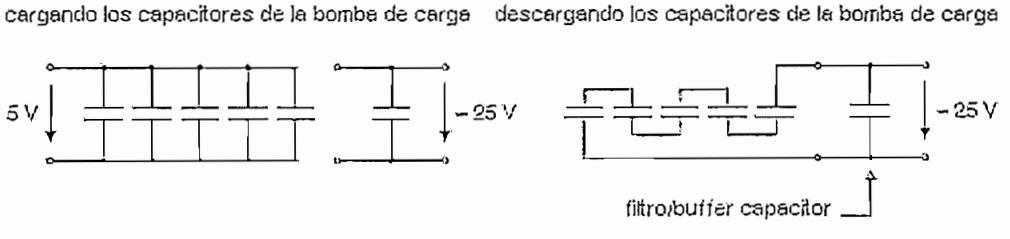

Figura 2.25 Principio de operación de un circuito de una bomba de carga durante la carga (izquierda) y descarga (derecha). (*Referencia:* [WOLFGANG])

El estado de borrado de las celdas de la EEPROM es el factor crítico con respecto a la escritura en las celdas. En la mayoría de tipos de EEPROM, el estado de borrado es "1". Una EEPROM tiene la propiedad de que una celda individual puede solo ser programada desde su estado de borrado a su estado de

no borrado, el cual en este ejemplo es "0". Si una celda EEPROM ya está en estado "0", una página entera de la EEPROM debe ser borrada para restaurar ese bit al estado "1".

La EEPROM es uno de los pocos tipos de memoria de semiconductor que tiene un número limitado de ciclos de acceso. Puede ser leída cualquier número de veces, pero puede ser programada solo un número limitado de veces. La razón de esta limitación se halla en su estructura de semiconductores.

2.3.2.3 Memoria Flash EEPROM (flash electrically erasable programmable read-only memory)

La flash EEPROM, la cual a menudo es llamada como 'Flash memory', comparte las propiedades de las memorias no volátiles como las EEPROM regulares. Esto significa que mantiene los datos en ausencia de energía. Es muy similar en su construcción y operación a la EEPROM. La diferencia básica entre una Flash EEPROM y una EEPROM normal está en el proceso de escritura.

Existen algunos microcontroladores de TIs con Flash EEPROM, la cual es en principio usada en microcontroladores de TIs como un reemplazo de ROM programada. Usando un microcontrolador con Flash EEPROM se puede reducir el tiempo de desarrollo de un proyecto de TIs en muchos meses, por este motivo se eliminó la necesidad de generar plantillas para ROM.

Desafortunadamente, es extremadamente difícil hacer dispositivos de semiconductor que tengan EEPROM y flash EEPROM sobre el mismo chip. Consecuentemente, en la práctica un microcontrolador con Flash EEPROM usualmente no contiene ninguna EEPROM regular. En lugar de eso, la EEPROM es reemplazada por una Flash EEPROM de alrededor de 8kB, la cual tiene la página más pequeña posible para minimizar el impacto sobre el COS. Este tamaño de página de la flash memory usado para reemplazar es generalmente significativamente más grande, toda vez que las rutinas almacenadas en estas memorias son escritas solo rara vez. Cuando el chip es fabricado, un cargador de

auto arranque es almacenada en una pequeña ROM para permitir que el fabricante de la TI cargue el código de programa y datos dentro de la Flash EEPROM.

Las celdas de memorias Flash EEPROM actualmente utilizadas tienen un periodo de garantía de retención de datos de al menos 10 años, al menos 100000 ciclos de escritura/borrado y tamaños de página típicos de 8-128 bytes.

2.3.2.4 Memoria RAM (random-access memory)

En las TIs, la RAM es la memoria usada para mantener datos que son almacenados o alterados durante una sesión. El número de acceso es ilimitado, la memoria RAM necesita una fuente de energía para su operación. Si la energía es apagada o falla temporalmente, el contenido de la RAM será indefinido.

Una celda RAM consiste de algunos transistores, conectados tal que ellos trabajan como un multivibrador biestable. El estado de este multivibrador representa el valor almacenado de un bit en la RAM. La RAM usada en TIs es estática (SRAM), lo cual significa que su contenido no tiene que ser refrescada periódicamente. Esto es debido a que no depende de un reloj externo, en contraste a una RAM dinámica (DRAM). Es importante para la RAM ser estática, ya que debe ser posible parar la señal de reloj de una TI. Con RAM dinámica, esto podría causar que la información almacenada se pierda.

2.3.2.5 Memoria FRAM (Ferroelectric random-access memory)

FRAM es un nuevo desarrollo en tecnología de semiconductores. A pesar de su nombre, la FRAM no es volátil como la RAM, sino en lugar de eso retiene su contenido sin un voltaje de suministro. Este tipo de memoria saca provecho de las propiedades de los materiales ferroeléctricos para almacenar datos. La memoria FRAM es potencialmente ideal para memorias de las TIs, dadas sus muy deseables propiedades como medios de almacenamientos de datos. Solo 5

V son necesarios para la programación, los tiempos de programación están alrededor de 100 ns y el número máximo de ciclos de programación bordea el trillón. Sin embargo, las FRAM tienen dos desventajas. La primera es el limitado número de ciclos de lectura, lo cual hace que sean necesarios ciertos ciclos de refresco. La segunda, la cual es más significativa, es que la producción de FRAMs implica fases de elaboración que son difíciles de dominar.[12]

[12] *Referencia:* [WOLFGANG]

Capítulo 3

Sistemas operativos, protocolos de transmisión y comandos de las tarjetas inteligentes.

3.1 SISTEMAS OPERATIVOS

De acuerdo con el estándar alemán DIN 4430, un SO[1] no es más que el conjunto de programas de un computador digital que junto con las propiedades del sistema de cómputo forman la base para los posibles modos de operación del sistema, el cual controla y monitorea la ejecución de los programas. El SO de una TI está diseñado para trabajar con el interfaz serial bidireccional que se conecta al terminal.

A inicios de 1990, existían muy pocos SOs para TIs. Esto sucedió en parte debido a la memoria de capacidad muy limitada con la que contaban los procesadores en aquellos días. Por lo que el SO era diseñado para una determinada aplicación y no contaba con una librería de rutinas amplia en la ROM. La estructura de esos SOs era en su mayoría monolítica y solo podía ser modificada con un alto costo. Luego se empezaron a diseñar SOs estratificados (divididos en capas), en nuestros días aun mantienen esta estructura pero con innumerables refinamientos.

Uno de los primeros COS[2] fue STARCOS, este SO, que inició su desarrollo en 1990, habilitó muchas aplicaciones para ser almacenadas, usadas y administradas independientemente en una sola TI, incluso en aquellas tempranas fechas. Con el paso del tiempo, el término COS llegó a ser aceptado a través de todo el mundo como una designación para un sistema operativo de tarjetas inteligentes. Éste, a menudo forma parte del nombre del SO como en 'STARCOS'

[1] [SO] Sistema operativo
[2] [COS] Sistema operativo para TI

y 'MPCOS'. En la actualidad existen más de mil compañías que producen COSs de propósito general y de aplicación especifica.

3.1.1 EVOLUCIÓN HISTÓRICA DE SISTEMAS OPERATIVOS

Los COSs en su evolución han pasado por las mismas fases que los demás SOs. El programa original de propósito especial para una sola aplicación fue en varias ocasiones generalizado y extendido. El resultado final fue un SO estructurado y de propósito general que era fácil de usar.

Los programas para TIs, creados a inicios de los 80's, en el principio de su desarrollo no podían ser considerados como verdaderos SOs ya que no eran mas que programas de aplicación incrustados en la ROM del chip. Sin embargo, toda vez que la fabricación de microcontroladores programados es cara y demanda mucho tiempo, la demanda de rutinas kernel[3] de propósito general creció muy rápido. Los programas de aplicaciones de propósito especial almacenados en la EEPROM podrían ser construidos sobre estas rutinas, usándolas cuando sea necesario. Sin embargo, esto también incrementaba la demanda de memoria, lo cual condujo a algunas compañías a moverse en la dirección opuesta, de regreso a software especial para aplicaciones específicas. Los SOs de propósito general basados en comandos estándar son la norma, dado que en principio ellos pueden ser usados teóricamente por cualquier tipo de aplicación. Si esto no es posible por alguna razón, ellos al menos deben ser diseñados para poder ser modificados a fin de reunir los mínimos requerimientos de una aplicación particular con el mínimo de esfuerzo y en el menor tiempo.

El desarrollo histórico de los sistemas operativos puede ser ilustrado claramente si observamos las TIs usadas por la German Mobile Telephone Networks. La TI usada desde 1987 para C-netz tiene un SO optimizado para esta

[3] [Kernel] Corazón de un Sistema Operativo (Componente central)

aplicación. Las modificaciones incluyen un protocolo de transmisión customizado, comandos especiales y una estructura especialmente adaptada de archivos para la aplicación. Todo en todo, este tipo de tarjeta tiene ciertamente un SO completo, pero está diseñado a la medida de C-Netz, los componentes esenciales de la aplicación basada sobre este SO fueron hechos a la medida de los requerimientos especiales del SO orientado a librerías y su hardware base. El siguiente paso fue la transición de una solución especial a algo así como un SO de arquitectura abierta. Una representación de éste es la primera tarjeta GSM, cuyo diseño es significativamente más abierto y multifuncional. Cuando las TIs GSM fueron especificadas, existían ya estándares propuestos para los sets de comandos y estructuras de datos de las TIs, lo cual significa que la piedra angular fue buscar una compatibilidad entre varios SOs. Partiendo de esta base, los demás desarrollos vinieron paso a paso. Las aplicaciones basadas en estos SOs eran altamente independientes del hardware y se basaban en varias interfaces del SO.

Los SOs modernos para GSM tienen cualidades tales como administración de memoria, múltiples árboles de archivos y otras, lo cual les brindaba características muy cercanas a las que proveían los SOs multiaplicación. Ellos pueden administrar muchas aplicaciones independientemente mientras prevé la interacción entre ellas. La mayoría de ellos también tienen muy complejos estados de máquina y muy amplios sets de comandos y algunos de ellos soportan muchos protocolos de transmisión de datos. Un moderno SO para TIs debe ser capaz de correr la tercera parte del código de programa en la tarjeta. Todo SO moderno tiene una estructura dividida en capas, donde únicamente la capa mas baja depende del hardware.

3.1.2 PRINCIPIOS DE DISEÑO E IMPLEMENTACIÓN

Los errores de diseño se manifiestan primero en la fase de implementación y los mismos son un hecho de la vida en todos los proyectos de software. Para la

minimización de tales errores, es recomendable observar algunos principios durante el diseño e implementación de un **COS**.

Debido a su especificación funcional un **COS** es un **SO** seguro diseñado para administrar información y sobre todo mantenerla como confidencial. Además, no es posible hacer ningún cambio en software una vez que está en uso. El primer principio nace justamente de esta observación. Un **COS** debe ser extremadamente confiable, lo cual significa que debe tener un número extremadamente pequeño de errores. La eliminación total de error es algo nunca alcanzable, puesto que incluso los **COSs** mas pequeños son muy largos como para que todos sus procesos sean probados.

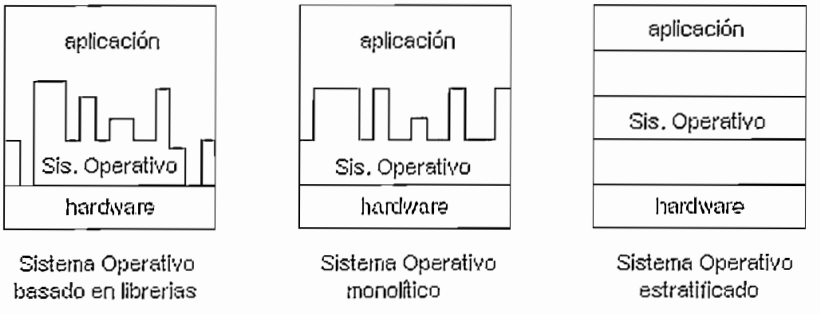

Figura 3.1 Representación esquemática del desarrollo histórico de los COSs. (*Referencia:* [WOLFGANG])

Sin embargo, un diseño estrictamente modular realiza una contribución decisiva para el descubrimiento y eliminación de cualquier error que podría presentarse en la fase de implementación. Esta modularidad, la cual robustece la confiabilidad del sistema operativo, no necesariamente resulta en un gran incremento del volumen de código de programación. Un beneficio adicional de un diseño modular es que cualquier caída del sistema que podría ocurrir, generalmente no afectaría la seguridad del sistema tanto como podría hacerlo en un código de programa altamente optimizado que ocupe menos espacio en memoria. Con un diseño modular, las consecuencias de cualquier error se mantienen localizadas y el SO como un todo es más robusto y estable.

El hecho que la mayoría de veces el software sea implementado en assembler hace que sea más propenso a errores. Un diseño basado en módulos individuales, completamente comprobables contribuye fuertemente a la detección de errores de programación de forma oportuna y limitando el alcance de sus efectos, debido al uso de interfaces definidas. La arquitectura de un sistema operativo dividido en capas que se observa en la Figura 3.1 es una consecuencia de esta aproximación modular. El gran esfuerzo en programación y planeamiento que esto requiere tiene una compensación, al menos financiera, por el hecho de que las pruebas y verificación son significativamente más fáciles. Por esta razón casi todos los sistemas operativos actuales tienen una arquitectura equivalente a alguno de los aquí descritos.

El desarrollo de software para el COS está cambiando desde la programación realizada enteramente en assembler. Muchos proyectos recientes han sido ejecutados desde un inicio en un lenguaje de alto nivel como C, el cual está relativamente cerca del hardware. Sin embargo, el kernel del sistema operativo está todavía basado en rutinas de assembler dependientes de la máquina, con todos los módulos de niveles más altos, tales como el administrador de archivos, los estados de máquina y el intérprete de comandos, programados en C. Esto reduce significativamente el tiempo de desarrollo y hace que el código sea más portable y reusable. Sobre todo, usando un lenguaje de alto nivel se mejora marcadamente el proceso de pruebas del software. Lo mejor es que una estructura de programa mas fácilmente entendible provista por un lenguaje de alto nivel también produce una tasa de errores claramente inferior.

Desafortunadamente, el programa generado por un compilador C, aun si éste es altamente optimizado, ocupa 20 o 40 % más espacio en la ROM que su equivalente código en assembler para la misma funcionalidad. Además, la velocidad de una implementación en C es marginalmente mas baja que una implementación en assembler. Sin embargo, ésta es solo crítica para los algoritmos de criptografía y los protocolos de transmisión de datos, dado que las

otras partes del software del COS normalmente no poseen procesos que dependen del tiempo de manera crítica.

El mayor problema de programar en C no es necesariamente el espacio extra de memoria que se necesita en la ROM o la reducida velocidad de ejecución. Sino la cantidad de RAM que es usada. Este tipo de memoria es muy limitada en las TIs, tiene su mayor desventaja en el hecho de tomar el mayor espacio por bit en el chip. Ésta es la razón principal por la que los lenguajes de alto nivel hasta ahora han sido usados más bien con moderación para programar COS.

Dado que las TIs son usadas en áreas de aplicación en las cuales la seguridad es un factor muy importante, el usuario de la tarjeta y/o el proveedor de la aplicación deben tener una confianza considerable en la integridad del productor del sistema operativo. Lo más reciente tiene cada oportunidad para tomar ventaja desleal del sistema entero por medio de aberturas de seguridad deliberadamente introducidas. Por ejemplo, consideremos una TI que se carga con cierto valor de dinero y que el comando para cargar dicha suma ha sido manipulado de tal forma que puede ser recargado sin autorización. Tales escenarios son la razón por la que pocos sistemas operativos han llegado a ser establecidos para su funcionamiento hoy en día. El riesgo de que un ostensible sistema operativo seguro contenga un caballo de Troya es significativamente más grande si éste viene de un pequeño y desconocido vendedor que si viene de una firma bien conocida en este campo. Sin embargo, recientemente ha habido un esfuerzo creciente para evaluar los COS de acuerdo al ITSEC o su sucesor, Criterio Común, para conseguir un nivel más alto de inteligibilidad y seguridad considerando tales escenarios.

3.1.3 ORGANIZACIÓN DE LA MEMORIA

Los tres tipos diferentes de memorias usados en microcontroladores de TIs tienen propiedades completamente diferentes. La ROM puede solo ser

programada usando un molde en el momento de la fabricación y se programa toda al mismo tiempo. Su contenido no cambia durante todo el tiempo de vida del chip. Dada la construcción de la ROM, la oportunidad de un cambio indeseado en el contenido de la ROM es prácticamente cero.

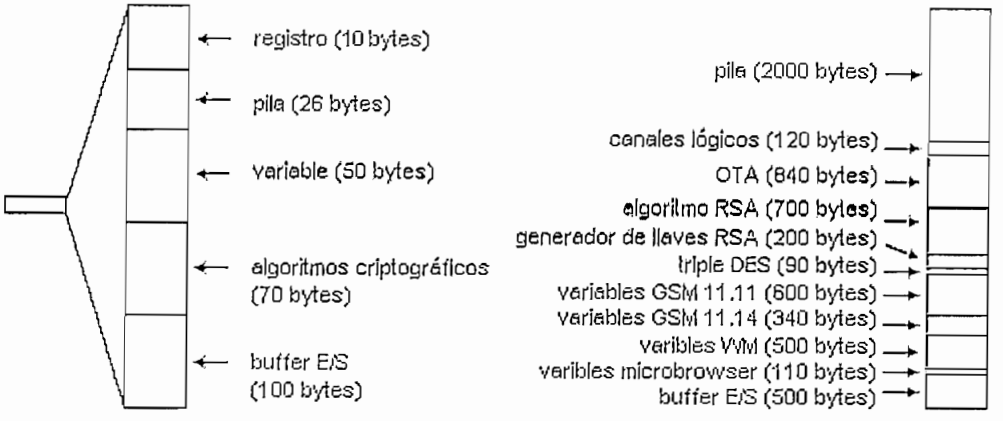

Figura 3.2 La figura de la izquierda muestra el particionamiento típico de una RAM de 256 bytes para un COS programado en assembler y la figura de la derecha la organización de la memoria de una RAM de 6 Kbytes para una TI de telecomunicaciones de alto rendimiento cuyo sistema operativo ha sido generado usando lenguaje C. (*Referencia:* [WOLFGANG])

En contraste a la ROM, la RAM guarda su contenido solo durante el tiempo que la TI está energizada. Una pérdida de energía causa la pérdida total de todos los datos almacenados en la memoria RAM. Sin embargo, se puede escribir en ella usando la velocidad completa de operación del procesador y puede ser borrada un número ilimitado de veces. La EEPROM, por su parte, puede mantener los datos sin necesidad de estar energizada. Sin embargo, ésta tiene tres desventajas, las cuales son su limitado tiempo de vida, el hecho de que está dividida en páginas y su relativa lentitud para ser escrita y borrada (alrededor de1 ms/byte). Con excepción de los vectores de interrupción, los cuales son prescritos por el microprocesador, el código de programa del sistema operativo almacenado en la ROM no tiene que cumplir con alguna estructura particular. Las rutinas individuales pueden de este modo ser enlazadas unas con otras en cualquier secuencia según se desee, aunque se hace un intento para limitar la distancia de los saltos para ahorrar espacio en memoria. Lo importante es que la ROM esté

protegida por medio de códigos de detección de errores (EDC), ya que ciertamente es posible que un error ocurra ocasionalmente en la ROM. Un rasguño en la región de la ROM del microcontrolador, por ejemplo o una fractura en el proceso de unión del alambre, pueden causar que los datos en la ROM sean incorrectos. Es muy interesante, que esto no necesariamente significa que el sistema operativo ya no puede funcionar; en lugar de eso, es ciertamente posible que solo rutinas específicas corran incorrectamente. Para prevenir problemas que podrían surgir en esas situaciones, la ROM es chequeada cuando el sistema operativo inicia su operación para verificar que esté fundamentalmente libre de errores.

La Figura 3.2 muestra la usual organización de una memoria RAM de 256 Bytes. Está dividida en regiones para los registros, el stack o pila, variables generales, espacio de trabajo para algoritmos de criptografía y el buffer de E/S. Si se requiere un buffer de 256 Bytes, por ejemplo o si variables adicionales deben ser almacenadas en la RAM, los límites de la memoria disponible pueden ser alcanzados muy de prisa. Este problema se resuelve teniendo el espacio de trabajo en la EEPROM, la cual es de este modo usada como RAM. La desventaja de esto es que toma casi 10000 veces escribir en la EEPROM de lo que tomaría escribir en la RAM. Una desventaja adicional es como se dijo antes el limitado tiempo de vida de las celdas de EEPROM, dado que son distintas a las celdas de la RAM, éstas no pueden ser borradas y escritas un número ilimitado de veces. Sin embargo, mover el contenido de la RAM a la EEPROM es a menudo la única solución, cuando se necesita que el buffer de E/S sea más grande de lo que es la RAM. Para efectos de comparación, en la Figura 3.3 también se muestra la organización típica de una memoria RAM de 6 KBytes en un TI para telecomunicaciones de alto rendimiento.

La organización de los datos almacenados en la EEPROM es mucho más complicada e intrincada que en cualquiera de los otros dos tipos de memorias. Con los sistemas operativos modernos, el particionamiento básico de una EEPROM es esencialmente como sigue (ver Figura 3.3). Primero, muchos tipos

de microcontroladores tienen una región al inicio de la EEPROM con una protección especial de hardware que puede ser usada para almacenar datos especiales de fábrica, tales como un número que se usa una sola vez y es por tanto único para el chip. Muchos fabricantes de semiconductores también registran en esta región el tipo de chip y la cantidad de EEPROM disponible para el sistema operativo. Esta región es usualmente designada para ser escrita una sola vez y leída muchas veces.

Figura 3.3 Ejemplo de la partición de una EEPROM para un COS. (*Referencia:* [WOLFGANG])

Sobre esta región, la cual usualmente tiene un tamaño de 16 a 32 bytes, vienen las tablas y punteros para el sistema operativo, los cuales son cargados dentro de la EEPROM cuando la tarjeta es finalizada. La combinación de aquellas tablas, punteros y rutinas almacenadas en la ROM hacen el COS completo. Para asegurar que el sistema operativo pueda ser usado siempre en un estado seguro y estable esta región también está protegida con un código de detección de errores (EDC) que es recalculado y chequeado antes del primer acceso a la EEPROM o antes de cualquier acceso a la EEPROM. Si un error en la memoria es detectado durante la verificación de EDC, la porción afectada de la EEPROM debe por consiguiente no ser usada. Dado que esto significa que la correcta operación del sistema operativo puede no estar asegurada.

Sobre la porción protegida del sistema operativo, existe una región que contiene código de programa de aplicación adicional. Si se necesita, esta región

también puede ser protegida contra alteraciones usando una suma de verificación. En esta región se colocan comandos de una aplicación específica o algoritmos que no deberían ser colocados en la ROM o que son muy largos para calzar correctamente en la ROM.

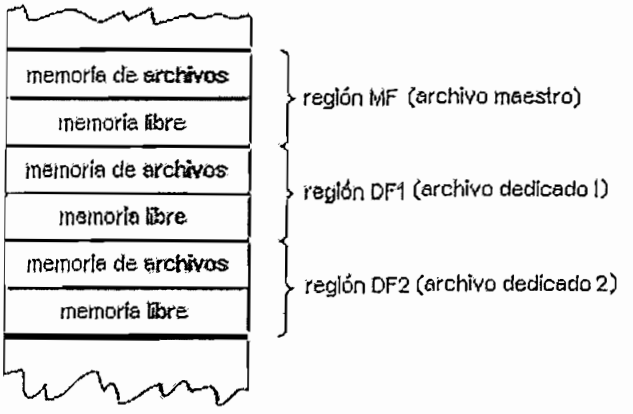

Figura 3.4 Particionamiento simple de la región de archivos de un COS que soporta algunas aplicaciones independientes y físicamente separadas. (*Referencia:* [WOLFGANG])

La siguiente región contiene toda la estructura de archivos o en otras palabras, el árbol de archivos visible externamente. Esta región no está protegida por completo por una suma de verificación, pero en lugar de eso tiene una protección muy fuerte basada en los archivos. La estructura de esta región se muestra en detalle en la Figura 3.4

3.1.4 ARCHIVOS DE LA TI

Además de contener mecanismos para identificación y autenticación, las TIs son primordialmente medios de almacenamiento de datos. Ellas tienen una ventaja decisiva en relación con otros medios de almacenamiento, tales como disquetes y está en el hecho de que el acceso a dichos datos puede estar sujeto a ciertas condiciones.

La primera TI tenía más o menos regiones de memoria direccionables directamente, las cuales podían ser usadas para escribir o leer datos. Los datos eran accesados por la especificación de direcciones físicas de memoria. Hoy en día, todas las TIs tienen sistemas completos de administración jerárquica de archivos con direccionamiento simbólico independiente del hardware.

Naturalmente, estos sistemas de administración de archivos tienen ciertas características que son específicas para las TIs. La característica más obvia es que no existe un interfaz hombre-máquina. Todos los archivos son direccionados usando código hexadecimal y todos los comandos están estrictamente basados en este direccionamiento, dado que aquí las comunicaciones solo toman lugar entre dos computadoras. Igualmente típico de estos sistemas de administración de archivos es que están diseñados para usar una pequeña cantidad de memoria. Se evita cada byte redundante si es posible, dado que el "usuario" en el terminal es una computadora esto no representa ningún problema.

Usualmente no existe ninguna forma de administración de memoria sofisticada, lo cual también es beneficioso para mantener el uso de memoria tan bajo como sea posible. Si un archivo se borra – y pocos sistemas operativos tienen esta capacidad – el espacio liberado no necesariamente viene a estar disponible para ser usado por un archivo recientemente creado. Normalmente, todos los archivos se crean y se cargan dentro de la TI cuando ésta se inicializa o personaliza. Después de esto, los cambios en el contenido de los archivos son limitados.

Naturalmente, las características de la memoria que se usa también afectará la naturaleza del sistema administrador de archivos. Las páginas de una memoria EEPROM no pueden ser escritas y borradas un número ilimitado de veces, como si se puede en el disco duro de una PC. Consecuentemente existen archivos con atributos especiales que permiten que la información sea almacenada en redundancia así ésta puede ser corregida si es necesario.

3.1.4.1 Estructura interna de los archivos

Los sistemas de administración de archivos modernos para **TIs** tienen una estructura orientada a objetos. Lo que significa que toda la información acerca de un archivo está almacenada en si mismo. Una consecuencia de este principio es que un archivo siempre debe ser seleccionado antes de que cualquier acción pueda ser realizada. Los archivos en tales sistemas orientados a objetos siempre son divididos en dos partes. La primera parte, se conoce como cabecera del archivo, contiene información acerca del diseño y estructura del archivo y sus condiciones de acceso. Los datos de usuario modificables se almacenan en la segunda parte, el cuerpo del archivo, el cual es enlazado a la cabecera del archivo por un apuntador o puntero.

Adicionalmente para proveer mejoras a la estructura de los datos, este esquema también tiene la ventaja de dar mejores seguridades físicas a los datos. La EEPROM orientada a páginas que contiene todos los archivos permite solo un limitado número de ciclos de escritura/lectura.

La cabecera y el cuerpo del archivo siempre son localizados en páginas de memoria diferentes. La cabecera, la cual normalmente es alterada pocas veces, almacena todas las condiciones de acceso. Un error al escribir o borrar que involucre el cuerpo del archivo de esta forma no afectará esas condiciones. Si el cuerpo y la cabecera del archivo fueran almacenadas en la misma página, esta podría ser posiblemente usada deliberadamente generando errores de lectura para alterar las condiciones de acceso tal que la información confidencial podría ser leída desde el cuerpo del archivo.

Algunos COS ofrecen la opción de direccionamiento del cuerpo de un archivo desde dos diferentes cabeceras. Esas dos cabeceras son usualmente localizadas en DFs[4] pertenecientes a dos diferentes aplicaciones. Esto permite

[4] [DF] Archivo dedicado

que los datos sean compartidos entre dos aplicaciones de una forma técnicamente estética. En este caso, es importante que las específicas condiciones de acceso en las dos cabeceras sean idénticas.

3.1.4.2 Tipos de archivos

La estructura de un sistema de archivos de TIs, como está especificado en el estándar ISO/IEC 7816-4, es similar al de un sistema DOS o Unix. La mayor diferencia es que las **TIs** no contienen ningún archivo de aplicación específica, tal como tipos especiales de archivos para un particular procesador de palabras. Solo la estructura de archivos estándar puede usarse en **TIs**.

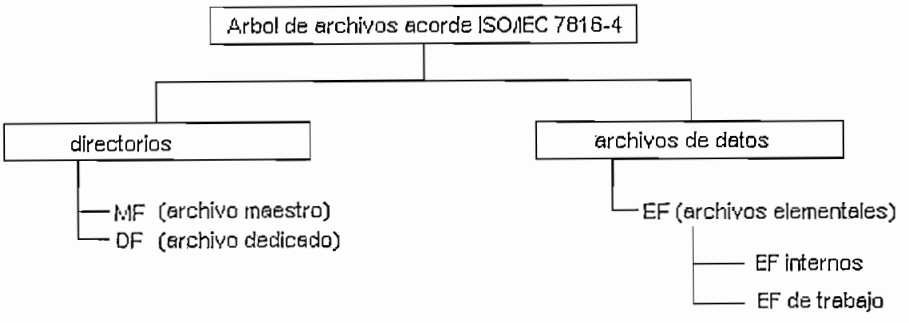

Figura 3.5 Clasificación de estructuras de archivos de TIs de acuerdo con el estándar ISO/IEC 7816-4. (*Referencia:* [WOLFGANG])

Para las **TIs** existen básicamente dos categorías de archivos. La primera categoría son los archivos directorio, los cuales se conocen como 'archivos dedicados' (DFs). La segunda categoría consiste en los archivos que mantienen los datos de usuario que están en uso en ese momento, los cuales se conocen como 'archivos elementales' (EFs). Un DF actúa como una especie de carpeta que contiene otros archivos, DFs o EFs de más bajo nivel que lógicamente permanecen juntos. EFs pueden ser clasificados en aquellos para el mundo externo (EFs de trabajo) y aquellos para el sistema operativo (EFs internos). La relación que existe entre los diferentes tipos de archivos se muestra en la Figura 3.6

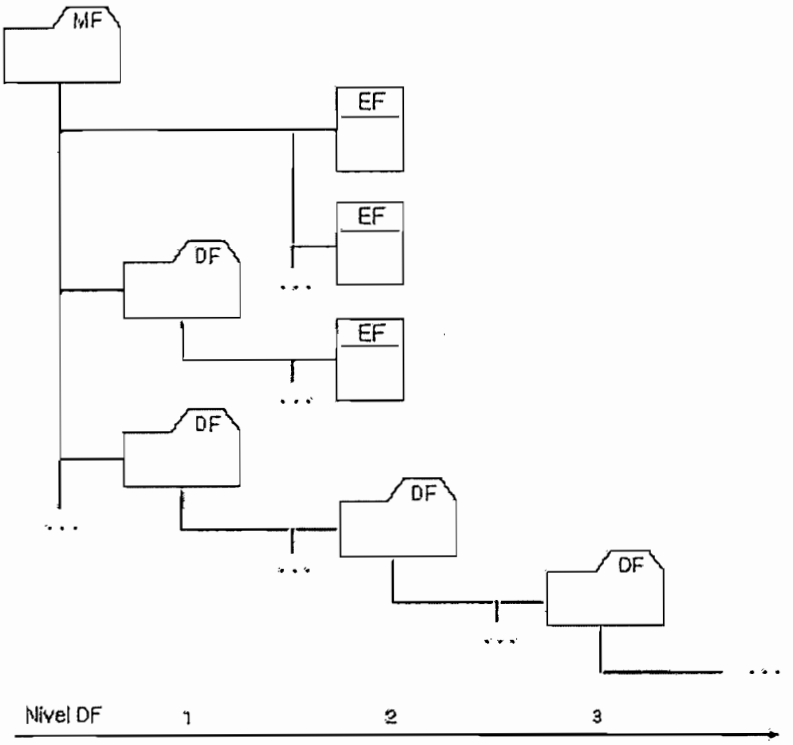

Figura 3.6 Los varios tipos de archivos en un árbol de archivos de una TI. (*Referencia:* [WOLFGANG])

3.1.4.2.1 Archivos MF

El directorio raíz es conocido como 'archivo maestro' (MF). Éste es seleccionado implícitamente después de un reset en la tarjeta. El MF contiene todos los demás directorios y archivos. Es un tipo especial de archivo DF y representa la extensión entera de la región de archivos disponible en la memoria de la tarjeta inteligente. Un archivo maestro debe estar presente en toda TI.

3.1.4.2.2 Archivos DF

Los archivos dedicados (DFs) podrían existir debajo del MF como se necesite. El término 'archivo directorio' es frecuentemente usado para estos archivos, aunque esto no cumple con la definición oficial de la abreviación 'DF' en el estándar ISO/IEC 7816-4. Un DF es un directorio en el cual otros archivos (DFs o EFs) pueden ser agrupados juntos. Un DF puede contener otros DFs. En principio, no existe límite sobre el número de niveles de DFs. Sin embargo, no es frecuente la existencia de más de dos niveles de DFs bajo el MF, debido a la

cantidad de memoria limitada en la TI. Con la especificación para la UICC (TS 102.221), un tipo especial de DF fue introducido con el nombre 'archivo dedicado de aplicación' (ADF). Este es un DF para aplicaciones y éste puede ser seleccionado usando un mecanismo apropiado (un comando SELECT con un AID o 'identificador de aplicación'), pero éste no está localizado debajo del MF. Un ADF puede de este modo ser considerado como un tipo de MF.

3.1.4.2.3 Archivos EF

Los datos de usuario necesarios para una aplicación están localizados en EFs. 'EF' es la abreviación de 'elementary file o archivo elemental'. Los EFs pueden estar ubicados directamente debajo del MF o debajo de un DF. Para permitir que los datos sean almacenados en una mínima cantidad de memoria y lógicamente optimizar la estructura de datos, un EF siempre tiene una estructura interna de archivo. Ésta es la principal diferencia entre EFs y archivos en un PC, cuya estructura de archivo está determinada por aplicaciones (como un procesador de palabras) en lugar de por el sistema operativo.

3.1.4.2.3.1 Archivos EF de trabajo

Todos los datos de aplicación deben ser leídos o escritos desde el terminal o en otras palabras, todos los datos que se desea sean para el mundo externo (visto por la TI), son localizados en los EFs de trabajo. Los datos que contienen tales archivos no son usados por el sistema operativo.

3.1.4.2.3.2 Archivos EF internos

Además de EFs para aplicaciones, existen también archivos internos de sistema que almacenan datos para el sistema operativo, datos para la ejecución de una aplicación, llaves secretas y código de programa. El acceso a estos datos está especialmente protegido por el sistema operativo. De acuerdo al ISO/IEC 7816-4, estos archivos de sistemas pueden ser integrados dentro del sistema de administración de archivos en dos formas diferentes. La primera opción es

almacenarlos en DFs de aplicación relevante como EFs invisibles. Tales archivos no pueden seleccionarse, ellos son administrados por completo y de forma transparente por el COS. Con la segunda opción, a esos archivos de sistema se les asigna un nombre regular (en otras palabras FIDs) y pueden ser seleccionados usando esos nombres.

Este es esencialmente el mismo principio que se aplica en la administración de archivos de DOS. Ambos métodos tienen aproximadamente igual número de ventajas y desventajas, con ambos proporcionando la misma funcionalidad de manera algo diferente.

3.1.4.2.4 Archivos de aplicación

De acuerdo con la convención, todos los archivos que contienen datos de usuario para una aplicación particular (los EFs para esa aplicación) están siempre juntos agrupados en un mismo DF. Esto da como resultado una estructura clara, entendible y facilita el ingreso de una nueva aplicación dentro de una **TI** por la creación del DF apropiado.

Toda vez que el MF es una clase especial de DF, esto no quiere decir que en una **TI** de Simple Aplicación, todos los archivos de aplicación puedan ser colocados directamente bajo el MF. En una **TI** típica de Simple Aplicación todos los EFs pueden ser colocados ya sea directamente bajo el MF o en un solitario DF. Las **TIs** con algunas aplicaciones tienen un correspondiente número de DFs, en los cuales se colocan los EFs pertenecientes a la aplicación.

Dentro de los DFs de aplicación DFs adicionales pueden ser colocados. Por ejemplo, un DF colocado directamente bajo el MF podría estar dedicado a una aplicación de 'Control de Tráfico', un nivel adicional de DFs con el DF de aplicación podría contener los archivos para los lenguajes soportados, tales como 'Ingles' y 'Español'.

3.2 TRANSMISIÓN DE DATOS EN TIS

La posibilidad de comunicaciones en dos vías es un prerrequisito para las comunicaciones entre una TI y un terminal. Sin embargo, solamente una línea está disponible (E/S), los datos digitales son intercambiados entre la tarjeta y el terminal vía esta conexión eléctrica. Dado que existe solamente una línea, la tarjeta y el terminal deben turnarse para la transmisión de datos, con la otra parte actuando como el receptor. Esta transmisión y recepción alternada de datos se conoce como transmisión half dúplex. Una transmisión full dúplex, en la cual ambas partes pueden transmitir y recibir datos simultáneamente, no se halla implementada en el presente para las TIs. Sin embargo, dado que la mayoría de procesadores de TIs tienen dos puertos E/S y dos de los ocho están reservados para aplicaciones futuras (tales como un segundo contacto E/S o una interfaz USB), la operación full dúplex podría ciertamente ser técnicamente posible. Esto indudablemente será implementado en el hardware y sistemas operativos en un futuro próximo.

La comunicación con la tarjeta siempre se inicia en el terminal, la tarjeta, por tanto siempre responderá a los comandos del terminal. Lo cual significa que la tarjeta nunca envía datos sin un estímulo externo. Esto deja entrever que existe una relación puramente maestro-esclavo, con el terminal como maestro y la tarjeta como esclavo.

Después de que una tarjeta se ha insertado en el terminal, primeramente sus contactos se conectan mecánicamente a los del terminal. Los cinco contactos activos son luego habilitados eléctricamente con la secuencia correcta. Después de esto, la tarjeta automáticamente ejecuta un reset de encendido y entonces envía una Answer To Reset (ATR) al terminal. El terminal evalúa la ATR, la cual contiene varios parámetros relacionados con la tarjeta y la transmisión de datos y luego se envía el primer comando. La tarjeta procesa los comandos y genera una

respuesta, la cual se envía al terminal. Esta interacción de comandos y respuestas de un lado para el otro continúa hasta que se desactiva la tarjeta.

Entre la ATR y el primer comando enviado a la tarjeta, el terminal puede también enviar un comando de Parámetro de Selección de Protocolo (PPS). El terminal puede usar este comando, el cual como la ATR es independiente del protocolo de transmisión, para poner varios parámetros de transmisión relacionados con el protocolo de transmisión de la tarjeta.

El procedimiento completo para la transmisión de datos a y desde la TI puede ser representado usando el modelo de capas OSI. Esto diferencia los eventos eléctricos en la línea E/S, procedimientos lógicos en el actual protocolo de transmisión y el comportamiento de aplicaciones que usan estos procedimientos. El comportamiento e interacción con y entre estas capas se especifica en algunos estándares internacionales. Estas relaciones se ilustran en la Figura 3.7

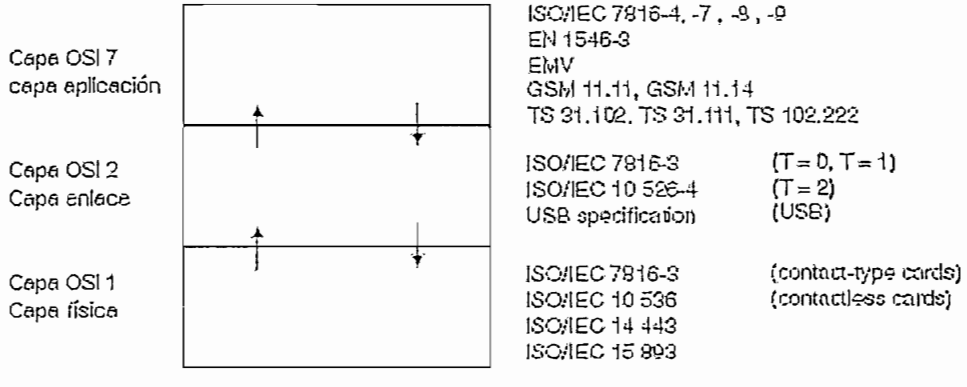

Figura 3.7 El modelo OSI para comunicaciones entre el terminal y la TI. (*Referencia:* [WOLFGANG])

En la práctica a menudo sucede que las TIs no soportan todas las opciones de protocolos de transmisión, dada la limitada memoria disponible.

Desde una perspectiva funcional, las diferentes opciones pueden ser un simple rango de posibilidades desde las cuales un óptimo conjunto puede ser seleccionado para una aplicación o TI particular. La consideración importante es que los parámetros seleccionados no deberían ser muy exóticos, para permitir que la tarjeta se comunique con tantos tipos de terminales diferentes como sea posible. En los terminales, cuando se aprecia la situación de los protocolos de transmisión de datos es algo diferente. En ellos la totalidad de funcionalidades relacionadas con los estándares relevantes está implementada, dado que se cuenta con suficiente memoria.

3.2.1 ANSWER TO RESET (ATR)

Después del suministro de voltaje y de que las señales de reloj y de reset han sido aplicadas, la TI envía un ATR vía la línea E/S. Esta cadena de datos, la cual contiene a lo mucho 33 bytes, siempre se envía con un valor divisor (factor de conversión de tasa de transferencia de datos) de 372 de conformidad con el estándar ISO/IEC 7816-3. Este ATR contiene varios parámetros relacionados al protocolo de transmisión y a la tarjeta. Este valor divisor debería ser usado incluso si el protocolo de transmisión utiliza más tarde el ATR empleando un valor divisor diferente (por ejemplo 64). Esto asegura que un ATR desde cualquier tarjeta puede ser recibido siempre, independientemente de los parámetros del protocolo de transmisión utilizado últimamente.

Es muy raro que un ATR tenga una longitud máxima permitida. A menudo consiste de solo unos pocos bytes. Particularmente en aplicaciones en donde se requiere que la tarjeta esté en uso rápidamente después de la secuencia de activación, el ATR debería ser particularmente corto. Un ejemplo típico es el pago de peaje usando una TI que se use como dinero electrónico. Si un vehículo pasa a través de la compuerta de peaje rápidamente, esto debe ser posible para debitar confiadamente de la tarjeta en el poco tiempo disponible.

El inicio de la transmisión de un ATR debe ocurrir entre 400 y 40000 ciclos de reloj después de que el terminal emita la señal de reset. Con una tasa de reloj de 3.5712 MHz, éste corresponde a un intervalo de 112 μs a 11.20 ms, mientras que con una tasa de reloj de 4.9152 MHz el intervalo es de 81.38 μs a 8.14 ms. Si el terminal no recibe el inicio de una ATR en este intervalo, éste repite la secuencia de activación varias veces (usualmente tres veces) para tratar de detectar un ATR. Si todos estos intentos no se logran, el terminal asume que la tarjeta está defectuosa y reacciona consecuentemente.

Durante el ATR, el tiempo entre los bordes mayores de dos bytes sucesivos puede ser hasta 9600 etu de acuerdo al estándar ISO/IEC 7816-3. Este período se diseña al 'tiempo de espera inicial' y es exactamente un segundo a una tasa de reloj de 3.5712 Mhz. Esto significa que el estándar permite un segundo de retardo entre bytes individuales del ATR cuando éste es enviado al terminal. En algunos COS, este tiempo se utiliza para cálculos internos y acceso de escritura a la EEPROM. El buffer interno de escritura para operaciones atómicas es a menudo excitado al mismo tiempo.

La cadena y elementos de datos del ATR están definidos y descritos en detalle en el estándar ISO/IEC 7816-3. El formato básico de un ATR está descrito en la Figura 3.8 y en la Tabla 3.1. Los primeros dos bytes, designados TS y T0, definen algunos parámetros de transmisión fundamentales e indican la presencia de subsiguientes bytes.

Tabla 3.1 Elementos de datos de la ATR y su asignación según el estándar ISO/IEC 7816-3. (*Referencia:* [WOLFGANG])

Elementos de datos	Designación
TS	Caracter inicial
T0	Caracter de formato
TA1,TB1,TC1,TD1,...	Caracteres de interfaz
T1,T2, ... , Tk	Caracteres históricos
TCK	Caracter de verificación

El caracter de interfaz especifica parámetros de transmisión especiales para el protocolo, los cuales son importantes para la transmisión de datos que está por venir. El caracter histórico describe la extensión de las funciones básicas de una TI. El caracter de chequeo, el cual es una suma de verificación de los bytes previos, podría opcionalmente ser enviado como el último byte de la ATR, dependiendo del protocolo de transmisión usado.

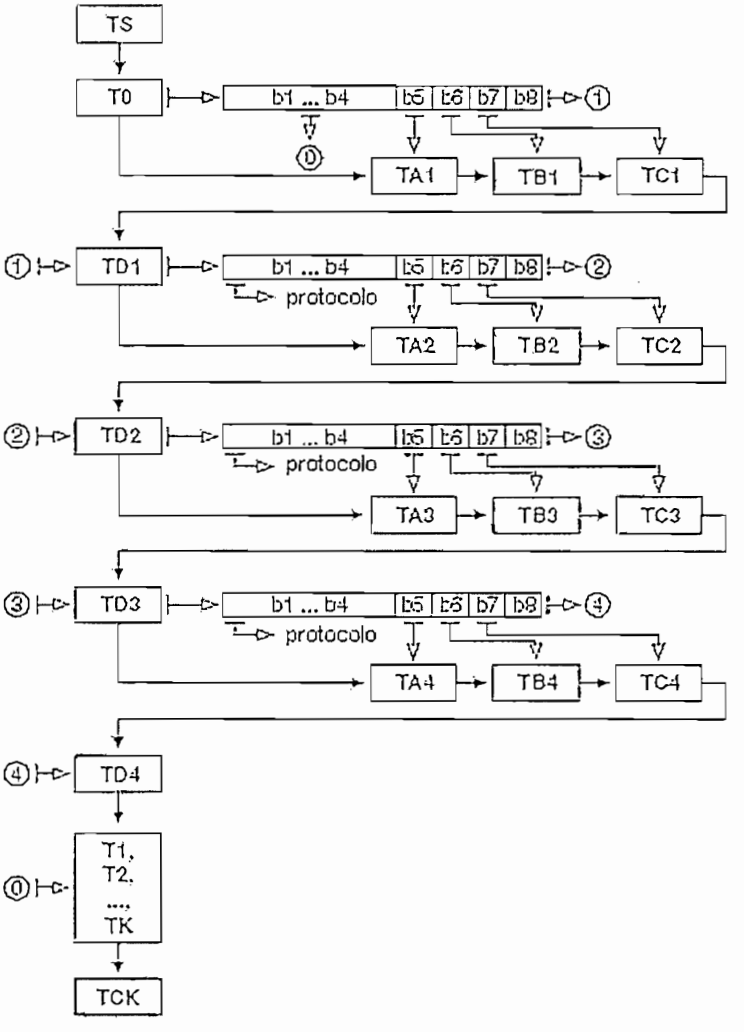

Figura 3.8 Estructura básica y elementos de datos de la ATR. (*Referencia:* [WOLFGANG])

3.2.2 PARÁMETRO DE SELECCIÓN DE PROTOCOLO (PPS)

La TI especifica varios parámetros de transmisión de datos en el carácter de interfaz del ATR, tales como el protocolo de transmisión y el caracter tiempo de espera. Si un terminal quiere modificar uno o más de estos parámetros, éste debe

ejecutar un procedimiento PPS de acuerdo con ISO/IEC 7816-3 antes de que el protocolo de transmisión se ponga en uso. Esto le permite al terminal variar ciertos parámetros del protocolo si es que es permitido por la tarjeta.

El PPS se puede realizar de dos formas diferentes. En el modo negociable, el valor estándar del factor *F* y el factor de ajuste de tasa de bit *D* se mantienen sin cambios hasta que se ejecute satisfactoriamente un PPS. Si la tarjeta usa el modo específico, el valor de F y D especificados en la ATR deben usarse para la transmisión de PPS. La tarjeta indica cuál de los dos modos soporta por medio del byte TA2.

La solicitud PPS debe ser hecha inmediatamente después de que el ATR ha sido recibido por el terminal. Si la tarjeta permite que se lleve a cabo los cambios solicitados a los parámetros de protocolo, se envía el PPS recibido de vuelta al terminal. En principio, éste es un eco del dato recibido. En otro caso, la tarjeta no envía nada y el terminal debe realizar una nueva secuencia de reset para hacer que la tarjeta salga de este estado. El PPS debe realizarse solo una vez, inmediatamente después de la ATR. Repetir la transmisión del PPS está prohibido en el estándar ISO/IEC 7816-3. En la práctica, rara vez se necesita ejecutar un PPS, dado que los parámetros de transmisión de las TIs que se usan están exactamente ajustados a los requerimientos del terminal.

3.2.3 PROTOCOLOS DE TRANSMISIÓN DE DATOS

Después de que una TI ha enviado un ATR, posiblemente seguido de un PPS, ésta espera por el primer comando del terminal. El proceso siguiente siempre corresponde al principio maestro-esclavo, con el terminal como maestro y la tarjeta como esclava. En términos concretos, el terminal envía un comando a la tarjeta, ésta lo ejecuta y a continuación envía una respuesta. Este ir y venir de comandos y respuestas nunca cambia.

Existen varias vías por las cuales la comunicación con una TI puede ser establecida. Existen también un número de métodos diferentes para la resincronización de comunicaciones si ocurriera una perturbación. La implementación exacta de los comandos, las correspondientes respuestas y los procedimientos usados en el evento de errores de transmisión están definidos en la forma del protocolo de transmisión. Un total de 15 protocolos de transmisión han sido identificados y definidos en términos de sus funciones básicas. Esos protocolos, los cuales son designados como 'T=' (por protocolo de transmisión) mas un número secuencial, están resumidos en la Tabla 3.2.

Tabla 3.2 Resumen de protocolos de transmisión de acuerdo con ISO/IEC 7816-3. (*Referencia:* [WOLFGANG])

Protocolo	Significado
T=0	Asincrónico, halfduplex, orientado al byte, especificado en ISO/IEC 7816-3
T=1	Asincrónico, halfduplex, orientado al bloque, especificado en ISO/IEC 7816-3 Enmienda 1
T=2	Asincrónico, full dúplex, orientado al bloque, especificado en ISO/IEC 7816-4
T=3	full dúplex, no especificado aun
T=4	Asincrónico, halfduplex, orientado al byte, extensión de T=0, no especificado aun
T=5...T=13	Reservado para uso futuro, no especificado aun
T=14	Para uso nacional, no estandarizado por ISO
T=15	Reservado para uso futuro, no especificado aun

Dos de esos protocolos predominan en el uso internacional. El primero es el protocolo T=0, el cual llegó a ser un estándar internacional en 1989 (ISO/IEC 7816-3). El otro es T=1, el cual fue introducido en 1992 en una enmienda a un estándar internacional (en aquel entonces ISO/IEC 7816-3 enmienda 1, hoy ISO/IEC 7816-3). El protocolo de transmisión full dúplex T=2, el cual está basado completamente en el protocolo T=1, está actualmente en la etapa de definición y será un estándar internacional en pocos años.

En Alemania, el ampliamente distribuido sistema de tarjetas telefónicas usa incluso un tercer protocolo, designado T=14. El cual está especificado en una especificación internacional de Deutsche Telekom.

Los elementos de datos transportados por protocolos de transmisión se llaman unidades de datos de protocolo de transmisión. (TPDUs). Ellos pueden ser

apreciados como recipientes dependientes del protocolo que transportan datos a y desde la tarjeta. Los datos de la aplicación en uso se incrustan en esos contenedores.

Adicional a los técnicamente complejos protocolos de transmisión de las TIs, existe un conjunto de protocolos sincrónicos muy simples para las tarjetas de memoria. Ellos son típicamente usados con tarjetas telefónicas, fichas médicas y cosas similares. Sin embargo ellos no tienen mecanismos de corrección de errores.

3.2.4 TRANSMISIÓN DE DATOS SINCRÓNICA

La transmisión de datos sincrónica no se usa con TIs basadas en microcontroladores, dado que ellas se comunican con el terminal solo de forma asincrónica. Sin embargo, éste es el método estándar para las tarjetas de memoria, las cuales se usan en un gran número de aplicaciones tales como monedas electrónicas para teléfonos monederos. Este uso común justifica una descripción de la operación de la transmisión de datos sincrónica.

En las memory cards, la transmisión sincrónica de datos está estrechamente ligada con el hardware del chip y está diseñada para ser tan simple como sea posible. No existe separación de capas en el protocolo de transmisión, ni el direccionamiento lógico está presente, por tanto la aplicación en el terminal debe acceder directamente a las localidades de memoria en el chip. El protocolo permite que los datos almacenados en el chip sean direccionados físicamente y luego escritos o leídos. Esto significa que el proceso de transmisión de datos está también atado a las funciones de direccionamiento y administración de memoria.

Tampoco existen procedimientos para detección o corrección de errores durante la transmisión de datos, aunque debemos mencionar que dichos errores entre la tarjeta y el terminal ocurren muy de vez en cuando. Si la aplicación en el terminal no obstante detecta un error de transmisión, ésta debe volver a leer el área relevante en la memoria de la tarjeta. Todas esas restricciones sirven para permitir a los datos ser transmitidos entre la tarjeta y el terminal usando solo una pequeña cantidad de circuitería lógica.

Ya que la transmisión de datos sincrónica es usada solamente para mantener una transmisión de datos tan simple como sea posible (lo cual significa usar la mínima cantidad de circuitería lógica posible), una casi inevitable consecuencia es la fuerte dependencia del hardware. Como resultado, los protocolos de transmisión sincrónica no son uniformes y algunas veces varían mucho de un chip a otro. Solo el ATR está estandarizado, un terminal que tenga que comunicarse con varios tipos de memory cards debe incorporar algunos tipos de protocolos de transmisión sincrónica diferentes.

La designación exacta del tipo de transmisión de datos usado por las memory cards es 'transmisión de datos serial sincrónica'. Esto indica claramente las condiciones básicas que se aplican para este tipo de comunicaciones. Como con las comunicaciones asincrónicas, los datos son transmitidos entre la tarjeta y terminal en forma serial o bit por bit. Sin embargo, los bits son enviados sincronizadamente a una señal de reloj transmitida adicionalmente esto hace que los bits de inicio y parada no sean necesarios.

En el caso de una simple memory card, tampoco existe detección de errores de la información, lo cual significa que ni un bit de paridad ni suma de verificación son enviados. La baja probabilidad de errores de transmisión se debe a la muy baja tasa de reloj, el rango está entre 10kHz y 100kHZ. Dado que un bit es enviado por cada ciclo de reloj, una tasa de reloj de 20kHz mantiene una tasa de transmisión de 20 kbit/s. Sin embargo, la tasa de datos efectiva es más baja,

debido a la información adicional de dirección que debe también ser enviada en el caso de las memory cards. Para describir la transmisión de datos sincrónica para las memory cards de una manera entendible, necesitamos describir algunas características básicas de las memory cards. En su forma más simple, estas tarjetas tienen memorias que son divididas en dos partes, una región ROM que no se puede cambiar y una región EEPROM que puede ser escrita y borrada. Ambas regiones son direccionables por medio de bits, se puede escribir en ellas libremente y la EEPROM puede también ser escrita y borrada.

La relación maestro esclavo es aun más pronunciada en las memory cards que en las TI con microprocesador. Por ejemplo, el terminal completamente asume el control del direccionamiento físico de la memoria. La tarjeta como tal puede solamente bloquear globalmente ciertas áreas para impedir que sean borradas. Esto se controla por la lógica del cableado localizado delante de la memoria. Esta lógica también administra el muy simple proceso de transmisión de datos.

La transmisión de datos será ilustrada aquí usando una tarjeta telefónica que contiene un chip Infineon SLE4403 como ejemplo. La memoria en esta IC está orientada al bit, lo cual significa que todas las operaciones son ejecutadas en bits individuales. Otros tipos de chip podrían tener protocolos que difieren del protocolo descrito aquí. Sin embargo, los principios básicos de la transmisión de datos son los mismos para todas las tarjetas sincrónicas.

Aquí los datos son transmitidos usando tres líneas. La líneas de datos bidireccional es utilizada tanto por el terminal como por la tarjeta para el intercambio de bits. La línea de reloj transmite el reloj generado por el terminal a la tarjeta. Este reloj provee la referencia para la transmisión de datos sincrónica. La tercera conexión necesaria para la transmisión de datos es la línea de control. Ésta determina lo que el chip hace, basado en los estados de las otras dos líneas.

En principio, el control de una memory card requiere la circuiteria lógica del chip para decodificar cuatro funciones diferentes. Aquellas son *leer, escribir, limpiar memoria e incrementar el apuntador de dirección.* Una memory card tiene un puntero global de memoria que se puede usar para direccionar cualquier región de la memoria bit por bit. Si el puntero alcanza el límite superior de la memoria, éste vuelve a cero. Con un chip diseñado para ser orientado al bit, este apunta al primer bit en memoria. Una de las funciones de la transmisión de datos sincrónica es la de resetear el apuntador a un valor inicial, el cual normalmente es cero. La siguiente función es para leer datos de la memoria. Las otras dos funciones son para escribir y borrar los bits de la EEPROM. El borrado de los bits de la EEPROM, podría permitir que se vuelva a escribir en ella, esto está por supuesto bloqueado en las tarjetas telefónicas, debido a que en otro caso las tarjetas podrían ser recargadas.

3.2.5 PROTOCOLO DE TRANSMISIÓN T=0

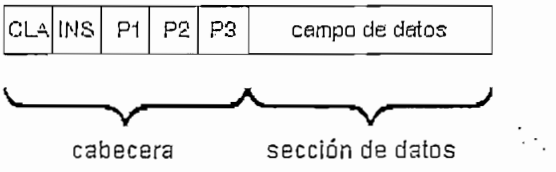

Figura 3.9 Estructura de un comando con el protocolo T=0. (*Referencia:* [WOLFGANG])

El protocolo de transmisión T=0 fue primero usado en Francia durante el desarrollo inicial de las TIs y fue también el primer protocolo para TIs estandarizado internacionalmente. Fue generado en los primeros años de la tecnología de la TI y es así designado para un mínimo uso de memoria y máxima simplicidad. Este protocolo se usa ampliamente en tarjetas GSM, lo cual lo hace el más usado de todos los protocolos de TIs actuales. El protocolo T=0 está estandarizado en ISO/IEC 7816-3, otras especificaciones adicionales compatibles se encuentran en el GSM 11.11, TS 102.221 y las especificaciones EMV. El protocolo T=0 es orientado al byte, lo cual significa que la unidad más pequeña procesada por este protocolo es un byte. La unidad de transmisión de datos consiste de una cabecera que contiene un byte de clase, un byte de comando y tres bytes de parámetros, opcionalmente seguido por una sección de datos. La

longitud de los datos es proporcionada solo por el parámetro P3. Éste indica la longitud del comando o respuesta y está también especificado en el estándar ISO/IEC 7816-3.

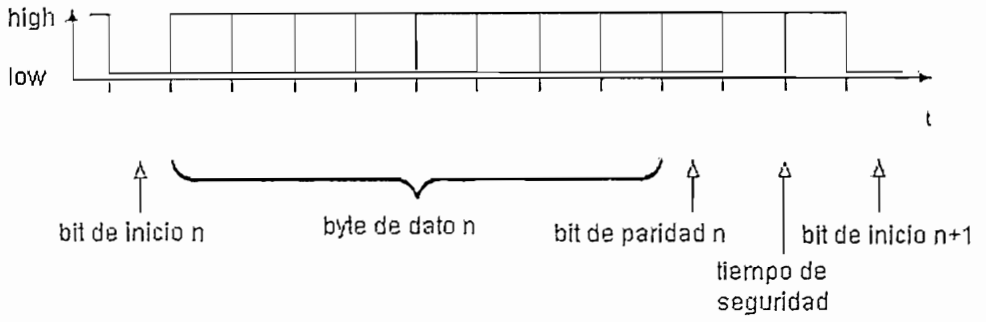

Figura 3.10 Un byte transmitido vía el interfaz E/S sin errores usando el protocolo T=0. (*Referencia:* [WOLFGANG])

Debido que el protocolo T=0 es orientado al byte, si un error en la transmisión es detectado, la retransmisión del byte errado debe ser solicitada inmediatamente. La detección de errores en el protocolo T=0 se hace mediante un bit de paridad, adjunto a cada byte enviado.

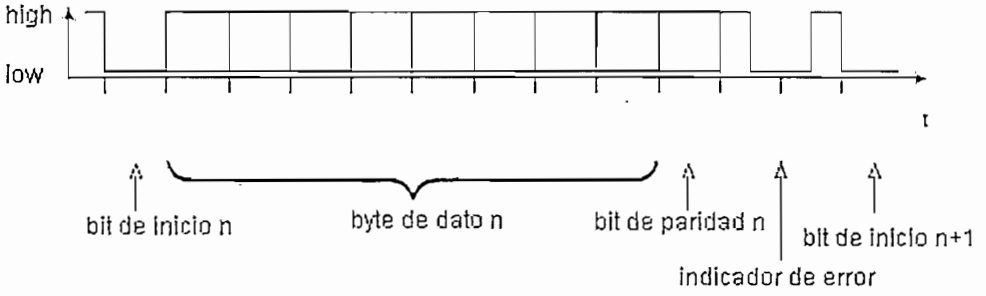

Figura 3.11 Un error de transmisión se indica en el protocolo T=0 por un nivel bajo en el interfaz E/S durante el tiempo de seguridad. (*Referencia:* [WOLFGANG])

Para ilustrar la secuencia comandos-respuestas, asumiremos que el terminal envía a la tarjeta un comando con una sección de datos y la tarjeta responde con datos y un código de respuesta (ver Figura 3.12). El terminal primero envía a la tarjeta la cabecera del comando formada por 5 bytes, que son un byte de clase, un byte de comando y los bytes P1, P2 y P3. Si éstos se reciben correctamente la tarjeta devuelve un ACK en la forma de un byte de

procedimiento (PB). Este byte ACK es codificado igual que el byte de comando recibido. Al recibir el PB, el terminal envía exactamente el número de bytes de datos indicados por el byte P3. Ahora la tarjeta ha recibido el comando completo y puede procesarlo y generar una respuesta.

Si la respuesta contiene datos adicionales a los dos bytes de código de respuesta, la tarjeta informa al terminal de esto mediante un código especial de respuesta, con la cantidad de datos indicada por SW2[5]. Después de recibir la respuesta, el terminal envía a la tarjeta un comando GET RESPONSE, el cual consiste solo de una cabecera de comando y una indicación de la cantidad de datos a ser enviados. La tarjeta ahora envía al terminal la solicitud de la cantidad de datos generada en respuesta al primer comando, con el código de respuesta apropiado. Esto completa una secuencia de comando. Si un comando es enviado a la tarjeta y ella solo genera un código de respuesta sin una sección de datos, la porción GET RESPONSE de la Figura 3.12 no ocurre.

Tarjeta Inteligente		Terminal
Cabecera recibida sin error, solicitud de transmisión de la sección de datos:[ACK] (acknowledge ACK)	← →	comando cabecera 5-byte [CLA,INS,P1,P2,P3]
procesamiento de comando Comando ejecutado y datos. disponible; longitud de datos en SW2: [SW1][SW2]	← →	[sección de datos] con número de bytes de datos = P3
[datos][SW1][SW2]	← →	GET RESPONSE P3=cantidad de datos a ser enviados [CLA,INS,P1,P2,P3] Secuencia comando-respuesta completa.

Figura 3.12 Una secuencia de comunicación típica T=0 con datos tanto en el comando como en la respuesta. (*Referencia:* [WOLFGANG])

[5] [SW1 y SW2] bytes de código de retorno

La función primaria de tiempo de seguridad es para separar bytes individuales durante la transmisión. Esto le da al receptor y al emisor más tiempo para realizar las funciones del protocolo de transmisión.

Si la TI retorna un byte de procedimiento conteniendo el valor nulo ('60') al terminal, éste no tiene ningún efecto sobre la secuencia actual del protocolo, pero informa al terminal que la TI está todavía procesando el último comando recibido. El envío de un valor nulo puede por lo tanto será usado como una especie de extensión del tiempo de espera (WTX), sin embargo no está estandartizado en esta forma.

El protocolo T=0 permite que la tarjeta reciba los bytes en la sección de datos individualmente después de que ha recibido la cabecera. Para hacerlo, solo debe enviar el byte comando invertido al terminal como un byte de procedimiento, después de lo cual el terminal enviará un byte de datos simple. El siguiente byte de datos sigue al nuevo byte de procedimiento de la tarjeta. Esta transmisión inteligente de datos puede continuar hasta que la tarjeta haya recibido todos los bytes en la sección de datos o hasta que ésta envíe al terminal el byte de comando no-inversión como un byte de procedimiento. Al recibir el byte de comando no-inversión el terminal envía a la tarjeta todos los bytes de datos faltantes, entonces habrá recibido el comando completo.

Con un protocolo de transmisión, las precauciones primarias del usuario son finalmente solo la tasa de transmisión de datos y los mecanismos de detección y corrección de errores. La transmisión de un byte de 8 bits requiere que se envíen 12 bits, incluyendo un bit de inicio un bit de paridad y dos etu para el tiempo de seguridad. La trasmisión de un byte por tanto toma 12 etu, lo cual es equivalente a 1,25 ms con una frecuencia de reloj de 3,5712 MHz y un valor divisor de 372. La Tabla 3.3 lista los tiempos de transmisión para algunos comandos típicos.

Tabla 3.3 Tiempos de transmisión para algunos comandos típicos usando el protocolo T=0. (*Referencia:* [WOLFGANG])

Comando	Datos de usuario	Datos de protocolo	Tiempo de Tx de datos
READ BINARY	C: 5 bytes R: 2+8 bytes	--	18,75 ms
UPDATE BINARY	C: 5+8 bytes R: 2 bytes	--	18,75 ms
ENCRYPT	C: 5+8 bytes R: 2+8 bytes	C: 5 bytes R: 2 bytes	37,50 ms

Naturalmente si ocurren errores en la transmisión la tasa de transmisión de datos cae. Sin embargo, el mecanismo de solo repetición de bytes es muy ventajoso en este punto, dado que solo los bytes recibidos incorrectamente han de ser retransmitidos. La detección de errores del protocolo T=0 consiste solo de un chequeo de paridad al final de cada byte. Dada la ausencia de separación entre capas y los obvios problemas en el evento de una mala conexión, el protocolo T=0 es a menudo considerado como obsoleto. Sin embargo, casi nunca ocurren errores en la transmisión entre la tarjeta y el terminal. Las principales ventajas del protocolo son sus buenas tasas de transmisión promedio, mínimos costos operativos de implementación y uso ampliamente generalizado.

3.2.6 PROTOCOLO DE TRANSMISIÓN T=1

El protocolo de transmisión T=1 es un protocolo asincrónico half-duplex para TIs, está basado en el estándar ISO/IEC 7816-3. Las especificaciones TS 102.221 y EMV también son relevantes para este protocolo, El protocolo T=1 es un protocolo orientado al bloque, lo cual significa que un bloque es la unidad de datos más pequeña que puede ser transmitida entre la tarjeta y el terminal.

Este protocolo se caracteriza por una estricta separación de capas. En este contexto, la separación de capas significa que los datos destinados para capas superiores, como la capa aplicación, pueden completamente ser procesados de forma transparente por la capa enlace. Sin que sea necesaria la intervención de

otras capas que no sean las que están directamente involucradas con la interpretación y modificación del contenido de los datos transmitidos.

Un intercambio de mensajes seguro (secure messaging SM), particularmente, requiere de la separación de capas. Solo entonces los datos encriptados del usuario serán pasados a través del interfaz sin tener que recurrir a procedimientos o trucos. El protocolo T=1 es el único protocolo de TIs que permite toda la variedad de transmisiones de datos segura sin ninguna componenda.

La secuencia del protocolo de transmisión empieza después de que la tarjeta ha enviado un ATR o después de que un PPS ha sido ejecutado satisfactoriamente. El primer bloque es enviado por el terminal y el siguiente por la tarjeta. La comunicación continúa de esta manera, con el privilegio de transmisión alternando entre la tarjeta y el terminal. Dicho sea de paso, el protocolo T=1 no se limita solo para comunicaciones entre TIs y terminales, sino que también es usado por muchos terminales para intercambiar datos de aplicación y control con las computadoras a las cuales están conectados. La tasa de transmisión es sin duda de interés para todo protocolo de transmisión de datos. La Tabla 3.4 muestra algunos tiempos de transmisión de comandos típicos usando el protocolo T=1.

3.2.6.1 Estructura del bloque

Los bloques transmitidos son usados básicamente para dos propósitos diferentes. Uno de los cuales es la transmisión transparente de los datos de una aplicación específica, mientras que el otro es el envío de datos para el control del protocolo o manejo de errores en la transmisión.

Un bloque consiste inicialmente de un campo de prólogo, un campo de información y un campo final de epílogo. Los campos de prólogo y epílogo son obligatorios y deben ser enviados siempre, el campo de información es opcional y

contiene los datos para la capa aplicación, el cual es ya sea una APDU comando enviado a la tarjeta o una APDU respuesta desde la tarjeta.

Tabla 3.4 Tiempos de transmisión para algunos comandos típicos usando el protocolo T=1. (*Referencia:* [WOLFGANG])

Comando	Datos de usuario	Datos de protocolo	Tiempo de Tx de datos
READ BINARY	C: 5 bytes	C: 4 bytes	28,75 ms
	R: 2+8 bytes	R: 4 bytes	
UPDATE BINARY	C: 5+8 bytes	C: 4 bytes	23,00 ms
	R: 2 bytes	R: 4 bytes	
ENCRYPT	C: 5+8 bytes	C: 4 bytes	38,75 ms
	R: 2+8 bytes	R: 4 bytes	

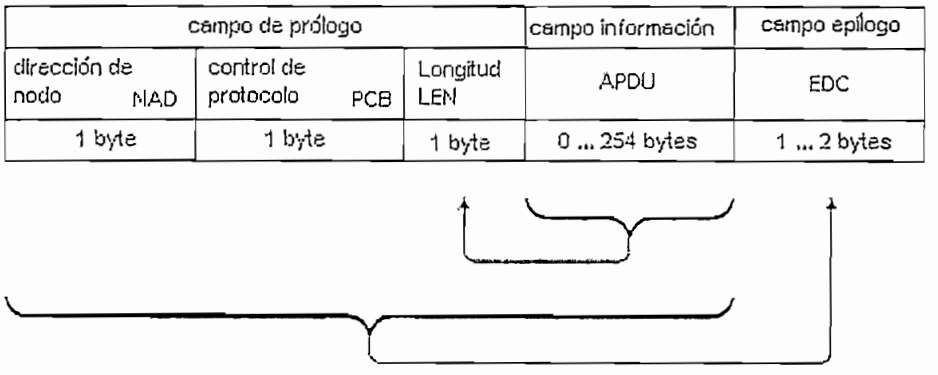

Figura 3.13 Estructura de un bloque transmitido protocolo T=1. (*Referencia:* [WOLFGANG])

Existen fundamentalmente tres tipos de bloques diferentes en T=1: bloques de información, boques acuse de recibo y bloques de sistema. Los bloques de información (bloques I) son usados para el intercambio transparente de datos de capa aplicación. Los bloques acuse de recibo (bloques R), los cuales no contienen ningún campo de datos, son usados para la confirmación positiva o negativa de la recepción de datos. Los bloques de sistema (bloques S) son usados para la información de control relacionada al protocolo. Según los datos específicos de control, pueden tener un campo de información.

3.2.6.2 Contador de secuencia de TX/RX

Cada bloque de información en el protocolo T=1 tiene un número de secuencia de envío que consiste solo de un bit localizado en el byte PCB. Esté contador es módulo 2, lo cual significa que puede alternar entre 0 y 1. El contador de secuencia de envío es también designado N(S). Su valor al iniciar el protocolo es 0. Los contadores en el terminal y en la tarjeta se incrementan independientemente uno del otro. El principal propósito del contador de secuencia de envío es soportar solicitudes de reenvío de bloques recibidos con errores, dado que bloques de datos individuales no pueden ser ambiguamente direccionados por medio del N(S).

3.2.6.3 Tiempos de espera

Algunos tiempos de espera están definidos para proveer a los transmisores y receptores intervalos máximos y mínimos específicos para varias acciones durante la transmisión de datos. Ellos también proveen formas definidas de terminar las comunicaciones con el afán de prevenir estancamientos en caso de errores. Los valores por defecto están definidos para todos estos tiempos de espera en el estándar, pero podrían ser modificados para maximizar la tasa de transmisión. Los valores modificados se indican en caracter específico de interfaz del ATR.

3.2.6.4 Caracter de tiempo de espera (CWT)

El caracter de tiempo de espera es definido como el máximo intervalo entre los límites delanteros de dos caracteres consecutivos en un bloque. El receptor inicia un timer de cuenta regresiva por cada límite delantero, usando el valor del caracter de tiempo de espera como valor inicial. Si el timer expira y ningún límite

delantero para un nuevo bit es detectado, el recipiente asume que el bloque transmitido ha sido recibido en su totalidad.

3.2.6.5 Bloque de tiempo de espera (BWT)

El propósito del bloque de tiempo de espera es permitir que las comunicaciones sean terminadas de una manera definida si la TI no responde. El bloque de tiempo de espera es el máximo intervalo permitido entre el limite delantero del último byte de un bloque enviado a la tarjeta y el límite delantero del primer byte retornado por la tarjeta.

En términos de un bloque convencional T=1, éste es el máximo intervalo permitido entre el límite delantero del byte XOR en el campo epílogo del bloque de comando y el límite delantero del byte NAD en la respuesta de la tarjeta. Si este período de tiempo espira sin que una respuesta sea recibida desde la tarjeta, el terminal podría asumir que la tarjeta está defectuosa e iniciar una contestación apropiada.

3.2.6.6 Bloque de tiempo de seguridad (BGT)

El bloque de tiempo de seguridad está definido como el mínimo intervalo entre el límite delantero del byte final y el primer byte en la dirección opuesta. Es el opuesto del BWT, el cual está definido como el máximo tiempo entre los dos límites delanteros específicos. Otra diferencia es que el bloque de tiempo de seguridad es obligatorio para ambas partes y debe ser observado, mientras el bloque de tiempo de espera es solo significativo para la tarjeta. El propósito del tiempo de seguridad es para proveer a quien envía los datos con un intervalo de tiempo mínimo en el cual pasa de transmisor a receptor.

3.2.6.7 Mecanismo del protocolo de transmisión

Si la tarjeta necesita más tiempo para generar una respuesta que el máximo tiempo permitido por el BWT, ésta puede pedir una extensión de tiempo de espera del terminal. Hace eso enviando un bloque S especial pidiendo una extensión y recibe un correspondiente bloque S del terminal en aceptación. El terminal no está habilitado para rehusar esta petición.

Un byte en el campo de información informa al terminal de la longitud de la extensión. Este byte, multiplicado por BWT da un nuevo BWT. Esta extensión es válida solo para los bloques I enviados recientemente.

Tarjeta nteligente		Terminal
Bloque S (petición de extensión de tiempo de espera)	← →	Bloque I
Bloque I	← →	Bloque S (aceptación de extensión de tiempo de espera)

Figura 3.14 Procedimiento para la extensión del tiempo de espera. (*Referencia*: [WOLFGANG])

3.2.6.8 Encadenamiento de bloques

Una de las características esenciales de desempeño del protocolo T=1 es la función de encadenamiento de bloques. Esto permite a cualquier parte enviar bloques de datos que son más largos que el tamaño de sus buffers de transmisión o recepción. Esto es particularmente útil en vista de la limitada capacidad de memoria de la TI. El encadenamiento es permitido solo para bloques de información, dado que solo tales bloques pueden contener alta cantidad de datos. En el proceso de encadenamiento, los datos de aplicación son particionados en bloques individuales que son enviados al receptor uno después de otro.

Los datos de capa aplicación deben ser particionados tal que ninguno de los segmentos resultantes sea más largo que el máximo tamaño del bloque del receptor. Entonces el primer segmento es colocado en un campo de información según lo acordado en el protocolo T=1, provisto con campos de prólogo y epílogo y enviado al receptor. El bit M (bit 'más datos') se pone en el campo PCB del bloque para indicar al receptor que la función encadenamiento de bloques está siendo usada y que datos encadenados están siendo colocados en los bloques que siguen. Tan pronto como el receptor ha recibido satisfactoriamente este bloque de información con el primer segmento de datos de usuario, indica que está listo para recibir el siguiente bloque I encadenado por medio del envío de un bloque R cuyo número de secuencia N(R) es el mismo del contador de secuencia de envío del siguiente bloque I. Luego el siguiente bloque es enviado al receptor. Este intercambio recíproco de bloques I y R continua hasta que el remitente emite un bloque I con un bit M en el campo PCB indicando que éste es el último bloque de la cadena (bit M=0). Luego de que este bloque ha sido recibido, el receptor tiene todos los datos de capa aplicación y puede procesar el bloque completo.

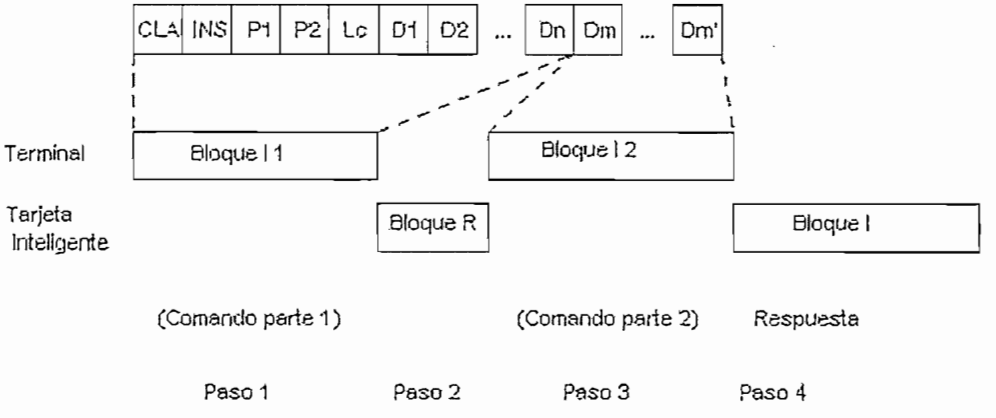

Figura 3.15 Ejemplo del encadenamiento de bloques para la transmisión de datos desde el terminal a la tarjeta. (*Referencia:* [WOLFGANG])

3.2.6.9 Manejo de errores

El protocolo T=1 tiene mecanismos y manejo de detección de errores elaborados. Si un bloque inválido es recibido, El protocolo intenta restaurar

unidad de datos estandarizada internacionalmente para la capa aplicación. En las TIs esta capa está localizada directamente sobre la capa protocolo de transmisión. Las unidades de datos dependientes del protocolo de la capa protocolo de transmisión se llaman 'transmission protocol data units' (TPDUs).

Una distinción se hace entre APDUs comando (C-APDUs), las cuales representan comandos a la tarjeta y APDUs respuesta (R-APDUs) las cuales son respuestas de la tarjeta a esos comandos. En términos simples, una APDU es una especie de contenedor que lleva un comando completo a la tarjeta o una respuesta completa desde la tarjeta. Las APDUs pasan por la capa protocolo de transmisión transparentemente, lo cual quiere decir sin modificación o interpretación.

Las APDUs que cumplen con el estándar ISO/IEC 7816-4 tienen por finalidad ser independientes de la capa protocolo de transmisión. Consecuentemente, el contenido y formato de una APDU no debe cambiar cuando se usa un protocolo de transmisión diferente. Esto se aplica sobre todo a los dos protocolos estándar, T=0 y T=1. Esta demanda de independencia del protocolo afecta la estructura de las APDUs, dado que debe ser posible transmitirla transparentemente usando tanto el protocolo orientado al byte T=0 y el protocolo orientado al bloque T=1.

3.2.7.1 Estructura de una APDU

Una APDU está compuesta de una cabecera y de un cuerpo. El cuerpo puede tener una longitud variable o podría estar completamente ausente si el campo de datos asociado está vacío.

La cabecera consiste de cuatro elementos, los cuales son el byte clase (CLA), el byte instrucción (INS) y dos bytes de parámetros (P1 y P2). El byte clase

se usa también para identificar aplicaciones y sus específicos sets de comandos. Por ejemplo el byte clase 'A0' es usado por GSM, mientras el código '8X' es usado por comandos de compañías especificas (uso privado). Los comandos basados en ISO son codificados por el byte clase '0X'. El estándar adicionalmente especifica los bytes clase usados para indicar el uso del intercambio de mensajes seguro y canales lógicos. Todo esto es aun compatible con el uso de un byte clase como un identificador de aplicación.

El siguiente byte en la APDU comando es el byte instrucción, el cual codifica el comando actual. Casi todo el espacio de dirección de este byte puede ser aprovechado, Esto es porque el protocolo T=0 permite al voltaje de programación ser accionado regresando el byte de instrucción incrementado en uno en el byte de procedimiento. De este modo el byte instrucción siempre ha tenido el valor par.

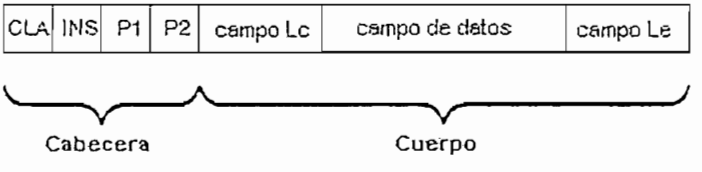

Figura 3.16 Estructura de una APDU comando. (*Referencia:* [WOLFGANG])

Los dos bytes parámetro son principalmente usados para proveer más información acerca del comando seleccionado por el byte instrucción. Dado que de este modo ellos funcionan como conmutadores que seleccionan varias opciones de comando.

La sección que sigue a la cabecera es el cuerpo, el cual puede ser omitido excepto para una especificación de longitud. El cuerpo realiza una función dual. Primero especifica la longitud de la sección de datos enviada a la tarjeta (en el campo Lc[6]) y la longitud de la sección de datos que va a ser enviada de regreso

[6] [Lc] longitud de comando

desde la tarjeta (en el campo Le[7]). Segundo, contiene los datos para el comando que son enviados a la tarjeta. Si el valor del campo Le es '00', el terminal espera que la tarjeta devuelva la máxima cantidad de datos disponibles para el comando. Ésta es la única excepción para la descripción numérica de la longitud.

Los campos Lc y Le usualmente son un byte largo, pero ellos pueden ser convertidos en campos que están cada tres bytes largos. Tales campos pueden ser utilizados para representar longitudes sobre 65536, dado que el primer byte contiene la secuencia de escape '00'. El estándar define estos tres bytes de especificación de longitud para futuras aplicaciones. Debido a las limitaciones de los actuales tamaños de memoria, éstas aun no han sido implementadas. Las partes previamente descritas de la APDU comando pueden ser combinadas para producir los cuatro casos generales ilustrados en la Figura 3.17

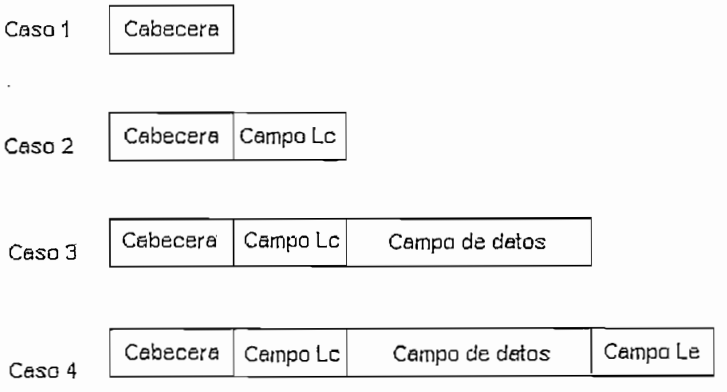

Figura 3.17 Los cuatro posibles casos de una APDU comando. (*Referencia:* [WOLFGANG])

3.2.7.2 Estructura de una APDU respuesta

La APDU respuesta, la cual es enviada al terminal en contestación a una APDU comando, consiste de un cuerpo opcional y un trailer obligatorio, como se muestra en la Figura 3.18. El cuerpo consiste del campo de datos, cuya distancia

[7] [Le] longitud de respuesta

está especificada por el byte Le de la APDU comando precedente. Independientemente del valor especificado en el byte Le, la longitud del campo de datos puede ser cero si la TI termina el procesamiento de comandos debido a un error o a parámetros incorrectos. Esto está indicado en los dos bytes simples, palabras de estado (status words) SW1 y SW2 en el trailer.

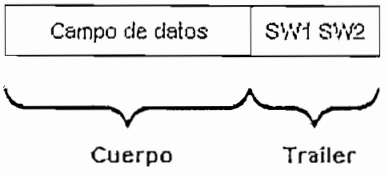

Figura 3.18 Estructura de una APDU respuesta. (*Referencia:* [WOLFGANG])

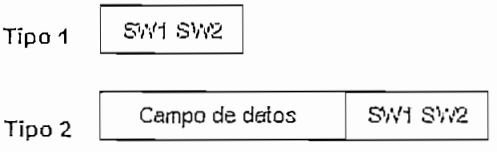

Figura 3.19 Los dos tipos de APDU respuesta. (*Referencia:* [WOLFGANG])

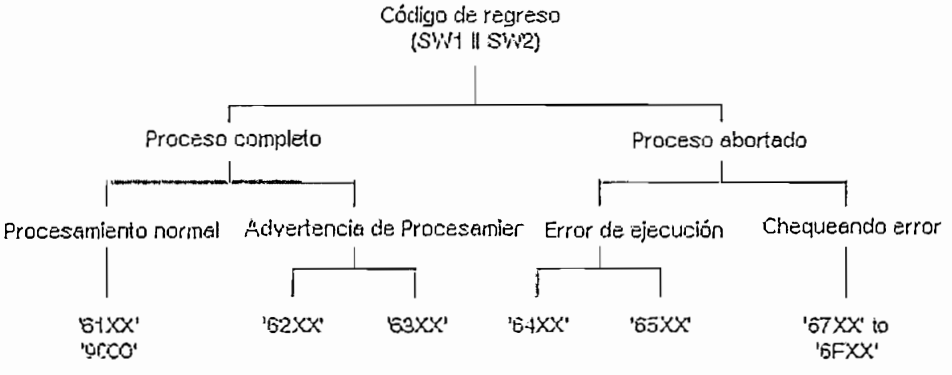

Figura 3.20 Esquema de clasificación del código de regreso como está definido en el estándar ISO/IEC 7816-4. Los códigos de regreso '63XX' y '65XX' indican que los datos en la memoria no volátil (EEPROM) han sido alterados, mientras que los restantes códigos '6X' indican que esto no ha ocurrido. (*Referencia:* [WOLFGANG])

La tarjeta siempre debe enviar un trailer en respuesta a un comando. Los dos bytes SW1 y SW2, los cuales también son llamados el 'codigo de regreso', codifican la respuesta al comando. Por ejemplo, el código de regreso '9000' significa que el comando fue ejecutado completa y satisfactoriamente. Existen

más de 50 códigos diferentes. Su esquema básico de clasificación se muestra en la Figura 3.20.

Si después de que un comando se ha ejecutado se recibe un código de regreso '63XX' o '65XX' esto significa que la memoria no volátil de la tarjeta (usualmente la EEPROM) ha sido modificada. Si es devuelto otro código que empiece por '6X', significa que la ejecución del comando terminó prematuramente sin la modificación de la memoria no volátil. Cabe notar que aunque existe un estándar para los códigos de regreso, muchas aplicaciones usan códigos que están estandarizados. La única excepción es el código '9000', el cual casi universalmente indica procesamiento satisfactorio. Con todos los demás códigos, es siempre necesario consultar la especificación relevante para asegurarse de sus significados.

3.2.8 TRANSMISIÓN DE DATOS SEGURA

El intercambio de datos completo entre el terminal y la TI usa pulsos eléctricos digitales sobre la línea E/S de la TI. Es concebible y no difícil técnicamente soldar un cable al contacto E/S, grabar todas las comunicaciones de una sesión y más tarde analizarla. En esta forma, es posible ganar conocimiento de todos los datos transmitidos en ambas direcciones.

Una tarea algo más difícil es aislar eléctricamente el contacto E/S, montar un contacto falso encima de él y luego usar cables delgados para conectar ambos contactos a una computadora. Con esta configuración, es fácil permitir que solo ciertos comandos alcancen a la tarjeta o insertar comandos 'exteriores' dentro de la secuencia de comunicaciones.

Estos dos tipos de ataques típicos pueden dar resultado sólo si los datos secretos pasan sobre la línea E/S sin protección. La transmisión de datos por lo

tanto debería ser diseñada tal que si algún atacante está en posición de escuchar a escondidas una transmisión de datos e insertar sus propios bloques de mensajes, esto no le permite tener ninguna ventaja.

Existen varios mecanismos y métodos que pueden ser usados para proteger las tarjetas de estos ataques e incluso de otros más sofisticados. Colectivamente nos referimos esos métodos como 'secure messaging = envío de mensajes garantizado'. Estos mecanismos no son específicamente para TIs y han sido usados por mucho tiempo en sistemas de comunicación de datos. Lo que es especial en el dominio de las TIs es que ni la capacidad de procesamiento de las partes que se comunican ni la tasa de transmisión es particularmente grande. Consecuentemente los métodos estándar comúnmente usados han sido llevados en una menor escala para hacerlos coincidir con las capacidades de las TIs, sin que de ninguna forma la seguridad de estos métodos se vea reducida.

El objetivo del envío de mensajes garantizado es asegurar la autenticidad y si es necesario la confidencialidad, de parte o toda la transmisión de datos. Una variedad de mecanismos de seguridad son usados para alcanzar este objetivo. Un mecanismo de seguridad se define como una función que requiere los siguientes ítems: un algoritmo de criptografía, una llave (key), un argumento y datos iniciales necesariamente. También una condición general debe ser satisfecha, la cual es que todo mecanismo de seguridad debe comportarse completamente transparente con respecto al protocolo existente de capas, para asegurar que procedimientos estandarizados no sean adversamente afectados por el envío de mensajes garantizado. Esto se aplica particularmente a los dos protocolos de transmisión T=0 y T=1, así como los comandos de TIs comúnmente usados.

Antes de usar un método de envío de mensajes garantizado, ambas partes deben estar de acuerdo en el algoritmo de criptografía que será usado y en una llave común. De acuerdo a los principios de Kerckhoff, la seguridad del método

depende por entero de esta llave. Si ésta es revelada, el envío de mensajes garantizado se reduce a una generalmente conocida suma de verificación adicional que decrementa la tasa de transmisión efectiva y a lo mejor puede ser usada para corregir errores de transmisión.

Con el paso de los años se han conocido algunos diferentes tipos de métodos de envío de mensajes garantizado. Ellos son relativamente rígidos y hechos a la medida de aplicaciones específicas. Sin embargo, ninguno de ellos ha llegado a ser internacionalmente predominante o ha probado ser suficientemente flexible para ser incluido en estándares actuales.

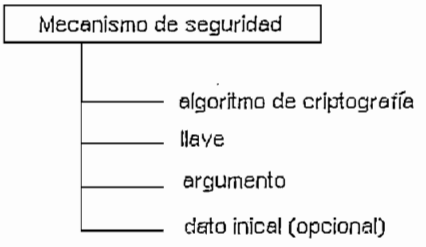

Figura 3.21 Datos y funciones requeridos para un mecanismo de seguridad. (*Referencia:* [WOLFGANG])

Los requerimientos de transparencia con respecto a los comandos existentes, usado con dos fundamentalmente diferentes protocolos de transmisión y máxima adaptabilidad ha conducido a la estandarización de un método de envío de mensajes garantizado muy flexible (correspondientemente complejo y elaborado) en el estándar ISO/IEC 7816-4, con funciones adicionales relatadas definidas en ISO/IEC 7816-8.

El byte clase indica si el envío de mensajes garantizado es usado por el comando. Los dos bytes disponibles pueden ser codificados si el método especificado en ISO/IEC 7816-4 se usa y si la cabecera está también incluida en la suma de verificación criptográfica (CCS).

3.2.8.1 Objetos de datos para texto plano

De acuerdo con el estándar, todos los datos que no están codificados con BER-TLV, lo cual significa que deben ser incrustados en objetos de datos (data objets). Algunas etiquetas diferentes son usadas, como se muestras en la Tabla 3.5 el Bit 1 de cada etiqueta indica si el objeto de datos está incluido en los cálculos de suma de verificación de criptografía. Si este bit no está puesto (Ej. 'B0'), el objeto de datos no está incluido en los cálculos, mientras que si está puesto (Ej. B1), el objeto de datos está incluido.

Tabla 3.5 Etiquetas para objetos de datos de texto plano. (*Referencia:* [WOLFGANG])

Etiqueta	Significado
'B0', 'B1'	Codificado BER-TLV; contiene objetos de datos relacionados con intercambio de mensajes seguro
'B2', 'B3'	Codificado BER-TLV; contiene objetos de datos no relacionados con intercambio de mensajes seguro
'80', '81'	Datos no codificados con BER-TLV
'99'	Información de estado para intercambio de mensajes seguro

3.2.8.2 Objetos de datos para mecanismos de seguridad

Los objetos de datos utilizados en mecanismos de seguridad están divididos en aquellos usados para autenticación y aquellos utilizados para confidencialidad. Las etiquetas definidas para este propósito se listan en las Tablas 3.6 y 3.7.

Tabla 3.6 Etiquetas para objetos de datos de autenticación. (*Referencia:* [WOLFGANG])

Etiqueta	Significado
'8E'	Suma de verificación criptográfica
'9A', 'BA'	Valor inicial para una firma digital
'9E'	Firma digital

Etiqueta	Significado
'82', '83'	Criptograma; el texto plano es codificado BER-TLV e incluye objetos de datos para intercambio de mensajes seguro.
'84', '85'	Criptograma; el texto plano es codificado BER-TLV y no incluye objetos de datos para intercambio de mensajes seguro.
'86', '87'	Indica el método de relleno usado: '01' – relleno con '80 00 ...' '02' – sin relleno

Aquí 'autenticidad' se refiere a todos los objetos de datos relacionados a sumas de verificación criptográficas y firmas digitales. La encripción de datos y el marcado de tales datos como encriptados en el contexto del intercambio de mensajes seguro, cae bajo el campo de 'confidencialidad'. Las etiquetas que se

Tabla 3.7 Etiquetas para objetos de datos de confidencialidad. (*Referencia:* [WOLFGANG])

Etiqueta	Significado
'82', '83'	Criptograma; el texto plano es codificado BER-TLV e incluye objetos de datos para intercambio de mensajes seguro.
'84', '85'	Criptograma; el texto plano es codificado BER-TLV y no incluye objetos de datos para intercambio de mensajes seguro.
'86', '87'	Indica el método de relleno usado: '01' – relleno con '80 00 ...' '02' – sin relleno

Aquí 'autenticidad' se refiere a todos los objetos de datos relacionados a sumas de verificación criptográficas y firmas digitales. La encripción de datos y el marcado de tales datos como encriptados en el contexto del intercambio de mensajes seguro, cae bajo el campo de 'confidencialidad'. Las etiquetas que se listan en las tablas anteriores deben ser usadas para un intercambio de mensajes seguro de acuerdo al tipo de método usado.

3.2.8.3 Objetos de datos para funciones auxiliares

Los objetos de datos para funciones auxiliares se usan en intercambio de mensajes seguro para coordinar restricciones generales. Las dos partes utilizan estos objetos de datos para intercambiar información acerca del algoritmo de criptografía y llaves utilizadas, el dato inicial e información básica similar. En principio, esos ítems pueden ser diferentes para cada APDU transmitida o aun entre un comando y su respuesta. En la práctica, sin embargo, los objetos de datos para funciones auxiliares raramente se utilizan, dado que todas las restricciones generales para el intercambio de mensajes seguro están definidas implícitamente, por tanto no tienen que ser definidas específicamente durante las comunicaciones.

Basado en las especificaciones para el intercambio de mensajes seguro especificado en ISO/IEC 7816-4, el cual ha sido solo brevemente esbozado anteriormente, podemos describir dos procedimientos fundamentales. El procedimiento de 'modo de autenticación' usa una suma de verificación

criptográfica (CCS o MAC) para proteger las APDUs contra manipulación durante la transmisión. El procedimiento de 'modo combinado', por otro lado, es utilizado para encriptar completamente los datos de aplicación, de modo que un atacante no pueda sacar ninguna conclusión acerca del contenido de los datos de comandos y respuestas que son intercambiados. Un contador de secuencia de envío solamente es utilizado con uno de esos procedimientos. Este contador, cuyo valor inicial es un número generado al azar, es incrementado por cada comando y cada respuesta. Esto permite a ambas partes determinar si un comando o respuesta ha sido omitido o insertado. Cuando un contador de secuencia de envío se usa en combinación con el procedimiento de 'modo combinado', APDUs idénticas aparecen como diferentes. Esto es conocido como 'diversidad'.

3.2.9 CANALES LÓGICOS

En TIs que contienen algunas aplicaciones independientes, es opcionalmente posible direccionar esas aplicaciones vía canales lógicos. Si los canales lógicos son usados, hasta cuatro aplicaciones en una sola tarjeta pueden concurrentemente intercambiar datos con el terminal. El interfaz serial existente es todavía usado, pero la aplicación puede ser direccionada individualmente a un nivel lógico.

Dos bits en el byte clase (bit 1 y bit 2) son usados para determinar cuál comando pertenece a cuál aplicación. Esto permite hasta cuatro canales lógicos, así hasta cuatro sesiones con aplicaciones en la tarjeta pueden correr en paralelo. Sin embargo, existe una limitación con respecto a la comunicación con varias aplicaciones en la TI, la cual es que los procesos externos que accedan a la tarjeta deben ser mutuamente sincronizados y no tiene permiso de interpaginar sus comandos, dado que la APDU respuesta de la tarjeta no contiene ninguna información acerca del canal lógico del comando que fue originado. Esto implica que es imposible determinar externamente cuál código de regreso ha sido

devuelto en respuesta a cuál comando. Debido a la ausencia de identificación de canal, nuevos comandos no pueden ser enviados hasta que la respuesta a los comandos previos haya sido recibida.

La aplicación primaria para este mecanismo muy poderoso es el hecho de usar algunas aplicaciones en paralelo. Por ejemplo, supongamos que una portadora de la tarjeta está llevando a cabo una conversación telefónica usando la aplicación GSM en una TI multiaplicación. Para confirmar un apunte con la otra parte, ella necesita brevemente consultar su organizador personal, el cual está localizado en la misma tarjeta. Usando un segundo canal lógico, el terminal busca un archivo en la aplicación organizador personal, en paralelo con la aplicación GSM, éste es el uso típico de los canales lógicos.

La utilidad potencial de los canales lógicos para aplicaciones es igualado por las dificultades de la administración de vínculos para el COS. En principio, cada canal lógico representa no menos que una TI separada, con todos sus estados y condiciones. Esto efectivamente significa que el SO debe concurrentemente administrar todos los datos para algunas sesiones paralelas con su memoria. Los costos asociados no deberían ser desestimados y en particular esto requiere microcontroladores con grandes cantidades de memoria. Si el envío de mensajes garantizado y todos los tipos posibles de autenticación son también requeridos para cada canal lógico, la cantidad de memoria necesaria crece muy rápidamente a un nivel que puede ser solo conocido por tipos de microcontroladores de TIs de más alto desempeño actualmente disponibles.

3.3 COMANDOS DE TARJETAS INTELIGENTES

Las comunicaciones actuales siempre emplean un protocolo de transmisión, tal como T=0 o T=1. Estos protocolos relativamente no complicados conocen los requerimientos especiales de una aplicación de TIs y están

optimizadas para este propósito. La desviación de estos protocolos precisamente especificados con procedimientos de aplicaciones no está permitida. Los protocolos de transmisión permiten a los datos ser transmitidos a la tarjeta y recibidos de ella de una manera completamente transparente a la capa transporte. Los datos son incrustados en una especie de contenedor llamado APDU. Las APDUS enviadas por el terminal a la tarjeta son los comandos. El terminal también recibe las respuestas a sus comandos en APDUs incrustadas en el protocolo de transmisión. Existe un gran número de comandos basados en este mecanismo y ellos inician actividades específicas con la tarjeta. Los ejemplos más simples de comandos son los de escritura y lectura de archivos de TIs.

En las aplicaciones de TIs, la tarjeta se usa como un medio para almacenar datos, un medio de autorización o ambos a la vez. Esto ha conducido a la generación de conjuntos de comandos optimizados para estas aplicaciones y protocolos de transmisión que se usan solo en el campo de las TIs. Debido a la cantidad de memoria severamente limitada de las TIs, combinada con la presión comercial para permitir solo incrementos moderados en esta capacidad por razones de costo, los conjuntos de comandos son usualmente hechos a la medida de aplicaciones específicas. Todos los comandos que no son necesarios para una aplicación dada son removidos inexorablemente durante la optimización del programa. Solo unos pocos COS muestran conjuntos de comandos extensos que no han sido reducidos a los necesarios para una aplicación específica.

Un efecto de diversificación es también visto con los sets de comandos de TIs, como típicamente sucede con nuevas tecnologías. Cada compañía activa en esta área intenta crear sus propios comandos que son hechos a la medida de las necesidades de sus SOs o de la aplicación. Esto a menudo resulta de la necesidad, toda vez que comandos de funcionalidad equivalente no podrían existir en los estándares. Las compañías también podrían deliberadamente intentar mejorar su posición relativa de competición o no permitir el acceso a sus competidores a áreas de aplicación particulares, utilizando comandos que son altamente optimizados con respecto a las funciones de la tarjeta y uso de

memoria. En cualquier caso, una decisión de usar comandos basados en estándares disponibles siempre significa escoger un sistema abierto, más fácilmente expandible y probado, el cual más tarde podría permitir que funciones adicionales sean incorporadas en una sola tarjeta. De otra parte, existen muchos ejemplos de sistemas en los cuales el uso de TIs fue solamente posible por el uso de comandos especiales altamente optimizados.

Actualmente existen 13 estándares internacionales y más o menos versiones borrador de estándares que definen comandos típicos de TIs. Ellos considerablemente definen más de 100 comandos, junto con sus procedimientos asociados. En gran medida, los comandos definidos son mutuamente compatibles en términos de codificación y funcionalidad.

La mayoría de comandos actualmente usados con TIs están definidos en el estándar ISO/IEC 7816-4. No está dedicado a ningún área particular, tal como telecomunicaciones o transacciones financieras, sino que en lugar de eso intenta direccionar todas las aplicaciones de TIs. Los comandos en ISO/IEC 7816-4 son complementados por tres secciones especializadas suplementarias de esta familia de estándares. ISO/IEC 7816-7 define comandos para la interrogación y administración de las bases de datos de TIs con estructuras basadas en lenguaje de interrogación estructurado (SQL). ISO/IEC 7816-8 contiene comandos para la ejecución y parametrización de funciones criptográficas y la parte 9 de la familia de estándares ISO/IEC 7816 suma comandos de administración de archivos al set de comando básico.

No existe un estándar internacional significante en el área de transacciones financieras, pero existe un estándar industrial. Ésta es la especificación EMV, de la que se habló anteriormente, debido a la fuerte posición en el mercado de las compañías detrás de esta especificación, la misma ha logrado posicionarse como una referencia para todos los COS y tiene el mismo grado de significancia que la familia de estándares ISO/IEC 7816.

La especificación GSM 11.11, la cual fue desarrollada para el uso en el área de telecomunicaciones, forma la base normativa para la SIM, mientras que los estándares TS 13.101, TS 13.111 y TS 102.222 especifican los comandos básicos para USIM. Dado el simple hecho de que un tremendo número de TIs se usan en aplicaciones de telecomunicaciones, estos estándares representan un estándar de facto para comandos de COS.

En principio, los comandos especiales usados solo en un área restringida no son cubiertos por estos estándares y deben por consiguiente ser especificados individualmente. Un ejemplo es el conjunto de comandos especificados para sistemas de monedero electrónico multiselector, el cual se define en el estándar CEN EN 1546. Este estándar europeo define todos los comandos necesarios para un monedero electrónico, junto con sus procesos asociados. Un estándar como éste, el cual está limitado a una sola aplicación, es tomado en cuenta en áreas de interés particular de agencias gubernamentales o ramas específicas de la industria, debido al costo muy elevado de generación de tales estándares.

Los comandos en los estándares y especificaciones descritos arriba pueden clasificarse de acuerdo a sus funcionalidades. Sin embargo, debemos recordar que solo subconjuntos de todos estos comandos son implementados en la vida real de los COSs. Las funciones descritas aquí están en principio presentes en todos los COSs. Por supuesto, la funcionalidad puede ser severamente restringida dada la consideración de cantidad de memoria o costo. Cuando sea que una aplicación nueva está siendo planificada, las especificaciones exactas de la codificación y funciones de todos los comandos deben ser solicitadas al o los productores del o los COSs bajo esta consideración.

Las siguientes secciones describen los comandos de TIs más importantes y más ampliamente usados. La base de esta selección se forma de los siguientes estándares y especificaciones: ISO/IEC 7816-4/7/8/9, EMV 2000, GSM 11.11, TS 311.111, EN 726-3 y EN 1546-3.

Naturalmente, es imposible comprar una sola TI en algún lugar del mundo que contenga todos los comandos existentes. Como una estimación conservativa, la memoria requerida para su completa implementación podría ser cinco o 10 veces más grande que la cantidad total de memoria del microcontrolador más grande de TIs actualmente disponible. Sin embargo, no es del todo necesario para una TI ser capaz de ejecutar todos esos comandos. Dependiendo del área de aplicación y COS deseados, ciertos comandos pueden ser soportados de forma más compresible que otros.

Algunos comandos son soportados por casi todos los COSs y tienen un limitado número de opciones. Para tales comandos, las APDU comandos y respuestas son completamente decodificados en su descripción.

Para una aplicación dada, los comandos de una TI pueden clasificarse en comandos operacionales, los cuales son necesarios para el uso normal y comandos administrativos, los cuales son necesarios para la administración de las TIs y las aplicaciones. Por razones relacionadas con la interoperabilidad, los comandos operacionales son generalmente especificados en completo detalle en los estándares. Los comandos administrativos son a menudo específicos para COS particulares y no necesariamente son especificados por los estándares.

Para cada comando, la respuesta mostrada en su descripción es la recibida por el terminal en el evento de una ejecución satisfactoria. En otro caso, si una operación es olvidada u ocurre un error en la tarjeta, el terminal recibe solo un código de regreso de 2 bytes. Algunos de los comandos descritos también tienen parámetros para seleccionar funciones adicionales. Estas opciones a menudo existen solo en el estándar, más que en el COS, toda vez que podrían ser muy complicadas o podrían no tener significancia práctica.

3.3.1 COMANDOS DE SELECCIÓN DE ARCHIVOS

Sin excepción, la administración de archivos de todo COS moderno está orientada a objetos. Entre otras cosas, esto significa que antes de que cualquier acción pueda ser realizada sobre un objeto (el cual corresponde a un archivo), este primero debe ser seleccionado. Solo entonces el sistema conoce de qué archivo se trata y todos los comandos subsiguientes específicos para el archivo se aplican solo a este archivo. Por supuesto, las condiciones de acceso para el archivo, aun deben ser chequeadas en el sistema operativo, para determinar si el comando en cuestión está permitido.

El archivo maestro (MF) siempre es implícitamente seleccionado después de que la tarjeta ha sido reseteada, así éste no sea específicamente seleccionado. Otros archivos son subsiguientemente seleccionados por la ejecución del comando SELEC FILE. Un archivo es seleccionado usando su identificador de archivo de 2 bytes (FID) o, en el caso de un archivo de directorio (DF), un nombre de DF de 1 a 16 bytes. Un nombre de DF puede contener un identificador de aplicación único internacionalmente (AID) cuya longitud está entre 5 y 16 bytes. Es posible pasar solo una porción del AID, omitiendo los bytes menos significantes (aquellos de la derecha). Un parámetro adicional hace que la tarjeta seleccione el primer, último, siguiente o previo DF relativo identificado por el AID abreviado.

En conexión con una definición de set de comandos más vieja, el estándar GSM 11.11 permite solo que 2 bytes FID sean usados para la selección de archivos. El set de comandos ISO, en contraste, también soporta un tipo de selección de archivos usando el nombre del camino del archivo en cuestión. El nombre del camino puede ser ya sea relativo, en cuyo caso el archivo es seleccionado a partir del DF actualmente seleccionado o absoluto, en cuyo caso el archivo es seleccionado a partir del MF.

Solo la selección satisfactoria de un nuevo archivo causa que el archivo seleccionado previamente sea deseleccionado. Si la selección no puede ser completada, por ejemplo por que el archivo seleccionado no existe, la selección previa se mantiene forzosamente. Esto asegura que un archivo está siempre seleccionado, incluso en el evento de un error.

Tabla 3.8 Funcionalidad del comando SELECT FILE acorde el estándar ISO/IEC 7816-4. (*Referencia:* [WOLFGANG])

SELECT FILE	
Comando	* FID (si EF, DF o MF) o Nombre DF (si DF) O Camino al archivo desde el DF seleccionado en ese momento O Camino al archivo desde el MF O Switch: Selecciona el siguiente DF de nivel mas alto O Primer, último, siguiente, anterior DF (Si un AID parcial es transmitido) * Switch: Devuelve información acerca del archivo seleccionado
Respuesta	* información del archivo seleccionado * código de retorno

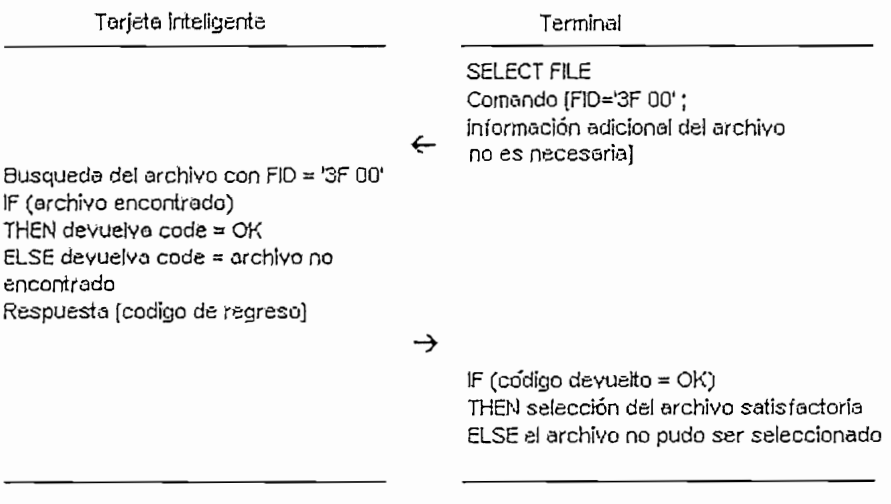

Figura 3.22 Secuencia de prueba del procesamiento del comando SELECT FILE. (*Referencia:* [WOLFGANG])

Después de que se ha seleccionado satisfactoriamente un archivo, el terminal podría solicitar información acerca del nuevo archivo actualmente seleccionado si es necesario. Esta solicitud, incluyendo el número deseado de ítems de datos, se envía a la tarjeta usando el comando SELECT FILE. El contenido exacto de esos ítems de datos está definido en el estándar. Los ítems de datos devueltos por la tarjeta podrían incluir información acerca de la estructura, tamaño y cantidad de memoria libre del archivo recién seleccionado. La cantidad de datos podría también depender del tipo de archivo. La Tabla 3.8 lista las opciones de selección de archivos explicita permitida por ISO/IEC 7816-4 para el comando SELECT FILE y la Figura 3.22 describe la secuencia de eventos en un típico proceso de selección de archivos.

3.3.2 COMANDOS READ AND WRITE

Los comandos read y write ante todo dan soporte al uso de TIs para el almacenamiento seguro de datos. Estos comandos pueden ser usados para escribir datos en los EFs apropiados y a continuación leer esos datos. Si esos EFs tienen condiciones de acceso específicas, solo a usuarios autorizados se les permitirá leerlos. Los datos son de esta manera almacenados en la tarjeta con protección contra accesos desautorizados.

Ya que existen diversos tipos de estructuras de datos EF, existen también varios tipos de comandos read y write para esos archivos. Desafortunadamente, éstos no corresponden completamente a un sistema de administración de archivos orientado a objetos. En un sistema puramente orientado a objetos, el SO debe ser construido de forma que un objeto pueda determinar sus propios mecanismos de acceso. Éste no es el caso para la administración de archivos en TIs. Esta no conformidad se remonta al surgimiento histórico de comandos que fueron subsiguientemente incorporados en los estándares actuales. Las precursoras de las TIs, las cuales son las memory cards, tienen solo una región de memoria que puede ser escrita y leída usando offset y parámetros de longitud.

Externamente, esta memoria puede ser apreciada como un solo archivo con una estructura transparente. Las primeras TIs fueron construidas de acuerdo al mismo principio y las definiciones de comandos para la lectura y escritura transparente de archivos data de este tiempo. Más tarde, cuando otros tipos de estructuras de archivos fueron definidos, nuevos comandos específicamente adaptados a esas estructuras fueron definidos para usarlos con tales archivos. Como resultado, existen dos tipos diferentes de acceso a los archivos.

Por esta razón esta clase de comandos debe ser dividida en comandos para acceder a EF con estructuras transparentes y comandos para acceder a EFs con otro tipo de estructura (cíclica, lineal fija y lineal variable). Sin embargo algunos estándares (tales como EN 1546 para monederos electrónicos) explícitamente establecen que los comandos de lectura de archivos con estructura transparente podrían también ser usados para leer archivos con otros tipos de estructuras. En cualquier caso, tales comandos pueden ser usados para obtener datos adicionales acerca de la estructura interna del archivo.

Un EF con una estructura lógica transparente es amorfo, lo que significa que no tiene ninguna estructura interna. Esto corresponde a un direccionamiento de memoria lineal. El comando RED BINARY se usa para leer tales archivos, mientras que los comandos WRITE BINARY y UPDATE BINARY se usan para escribir.

La diferencia fundamental entre los comandos WRITE BINARY y UPDATE BINARY se relaciona con el estado de seguridad de la EEPROM de la tarjeta. El estado de EEPROM seguro es el estado lógico de los bits de la EEPROM cuando las celdas de memoria han tomado sobre sí estados de energía mínima. Dado que las celdas de memoria son pequeños capacitores, esto significa que el estado en el cual ellas no contienen carga, lo cual es usualmente un estado lógico 0. Para cambiar un bit del estado 0 al estado 1, debe ser borrado, lo que restaura la carga del capacitor.

Un comando WRITE se puede usar solamente para cambiar bits de un estado no seguro, lo cual usualmente es un 1 lógico, al estado seguro, lo cual es usualmente un 0 lógico. En este caso el comando WRITE produce el AND lógico del dato transferido en el archivo contenido. En contraste, si el estado seguro del chip corresponde a un 1 lógico, el comando WRITE debe producir un OR lógico del dato provisto por el comando y el dato en el archivo. El resultado del acoplamiento lógico de los datos provistos por el comando y los datos en el archivo es que el estado seguro de la EEPROM siempre es activado usando un comando WRITE.

Un comando UPDATE, en contraste, realiza una escritura genuina al archivo. El estado previo de los datos en el archivo no afecta el contenido del archivo después de la ejecución de un comando UPDATE. Por tanto el comando UPDATE BINARY puede ser considerado como un equivalente a usar ERASE para borrar el archivo y luego el comando WRITE BINARY.

Estos comandos pueden ser usados para construir físicamente TIs contadoras. El principio involucra un campo de bit, en el cual cada bit colocado representa una unidad monetaria. Cuando se realiza un pago, el contador es decrementado bit por bit usando una operación OR generada por comandos WRITE BINARY. Después de la autenticación, el valor del contador puede ser incrementado otra vez usando un comando UPDATE BINARY. La ventaja principal de esta técnica es que ésta hace imposible que se incremente el valor del contador por manipulación de la EEPROM, como calentándola, dado que el estado seguro de cada bit representa un valor de 0.

Como su nombre lo sugiere, el comando READ BINARY es un comando de escritura, mientras que los comandos WRITE BINARY y UPDATE BINARY son comandos de escritura. El archivo siempre es accesado usando un parámetro de longitud y un offset al primer byte a ser direccionado. Algunos COSs también

permiten la selección implícita del archivo antes de que el acceso a los datos ocurra, usando un parámetro corto suplementario FID.

La Figura 3.23 ilustra una secuencia de comandos típica usando READ BINARY seguido de WRITE BINARY y finalmente UPDATE BINARY. Los efectos sobre el contenido del archivo seleccionado se muestran en la Figura 3.24. Por supuesto, este ejemplo muestra que la selección del archivo ha sido satisfactoria y que las condiciones de acceso para que el archivo sea leído son conocidas.

Smart card		Terminal
	←	READ BINARY *Comando* [offset = 2 bytes, número de bytes a ser leidos = 5]
datos demandados := '03' || 'FF' || '00' || 'FF' || '00' *Respuesta* [datos demandados] [código de regreso]	→	IF (código de regreso = OK) THEN READ BINARY satisfactorio ELSE abortar
	←	WRITE BINARY *Comando* [offset = 3 bytes, número de bytes a ser escritos = 2, dato = 'F0 F0']
Respuesta [código de regreso]	→	IF (código de regreso = OK) THEN WRITE BINARY satisfactorio ELSE abortar
	←	UPDATE BINARY *Comando* [offset = 5 bytes, número de bytes a ser escritos= 2, data = 'F0 F0']
Respuesta [código de regreso]	→	IF (código de regreso = OK) THEN UPDATE BINARY satisfactorio ELSE abortar

Figura 3.23 Accesando un archivo con una estructura transparente. (*Referencia:* [WOLFGANG])

El comando ERASE BINARY es una excepción entre los comandos que operan sobre EFs transparentes. Esto no puede ser usado para escribir datos en el archivo, sino solo para borrar los datos desde un offset dado. Si no es colocado un segundo parámetro offset, el comando borra todos los datos al fin del archivo seleccionado. En este caso, el borrado de datos significa que la región de datos especificada en el comando se pone al estado borrado lógico. Este estado debe

ser definido separadamente para cada sistema operativo, dado que éste no puede ser el mismo estado de borrar físicamente la memoria.

(a)	'01'	'02'	'03'	'FF'	'CC'	'FF'	'00'	···	n
(b)	'01'	'02'	'03'	'F0'	'CC'	'FF'	'00'	···	n
(c)	'01'	'02'	'03'	'F0'	'CC'	'F0'	'F0'	···	n

Figura 3.24 Ejemplo de acceso de escritura a un EF con una estructura transparente usando la secuencia de comandos mostrada en la Figura 3.23. (*Referencia:* [WOLFGANG])

(a) contenido del archivo con READ BINARY
(b) contenido del archivo con WRITE BINARY
(c) contenido del archivo con UPDATE BINARY

Debido a que las estructuras lineales fijas, lineales variables y cíclicas de los EFs son fundamentalmente diferentes de una estructura transparente de EFs, están disponibles además de los comandos descritos arriba comandos especiales para esas estructuras de datos particulares. Todos estos archivos tienen estructuras orientadas a registros. Para acceso de escritura, la unidad más pequeña que puede ser direccionada en el campo de datos es un solo registro. Para el acceso de lectura, ya sea un registro entero o una parte de él podría ser leída, empezando por el primer byte del registro. Estas estructuras de archivos, las cuales transforman una memoria lineal unidireccional en una memoria que puede ser direccionada en dos dimensiones, dio paso a tipos de acceso que son significativamente más complejos que aquellos usados con una estructura transparente. En principio, todas las estructuras de datos posibles, pueden ser emuladas usando una estructura transparente, pero en casos específicos esto ha probado ser considerablemente más complejo que el uso de estructuras de alto nivel.

Después que un EF con una estructura orientada al objeto ha sido seleccionado, el COS crea un puntero del registro cuyo valor inicial es indefinido.

El valor de este puntero puede ser colocado usando un comando READ, WRITE, UPDATE RECORD o SEEK. El puntero del archivo en uso en esos momentos es guardado mientras el archivo este seleccionado. Después que una selección explicita o implícita de otro archivo es realizada satisfactoriamente, el valor del puntero nuevamente es indefinido.

Todos los comandos para archivos orientados a registros pueden usar un byte parámetro para especificar el tipo de acceso al archivo. El tipo básico es el acceso directo usando el número absoluto del registro deseado. Este tipo de acceso no altera el puntero de registro. El número del registro deseado se envía a la tarjeta y la respuesta lleva el contenido del registro en cuestión.

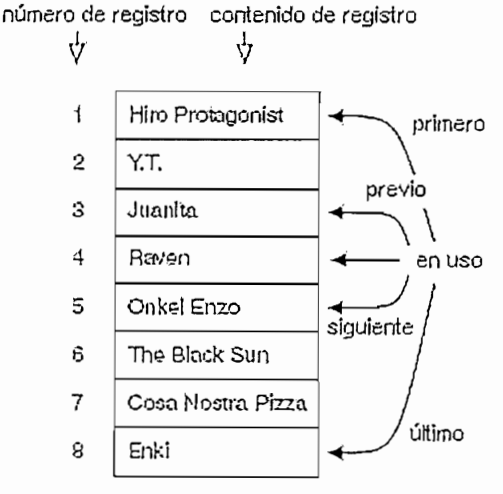

Figura 3.25 Accediendo a un archivo que tiene una estructura orientada al registro. (*Referencia:* [WOLFGANG])

Si el byte parámetro especifica el primer registro, el SO coloca el puntero de registro en el primer registro del archivo y este registro es escrito o leído según el tipo de comando usado. El valor del parámetro especificando el último registro accesa al registro final de manera similar. Los parámetros adicionales: 'siguiente' y 'previo' dan paso al registro siguiente y previo, respectivamente, para ser seleccionados y escritos o leídos. Finalmente el parámetro de valor 'actual' puede usarse para direccionar el registro marcado por el valor en uso en ese momento del puntero de registro.

Esta gran variedad de métodos de acceso para estructuras de datos orientadas a registros se origina de la típica estructura de un directorio telefónico. Considere un registro cuya parte inicial contiene el nombre y el apellido dado, seguido en el mismo registro por el número telefónico asociado. Usando READ RECORD y los parámetros arriba descritos con un directorio telefónico acotado en un EF, puede 'paginar' hacia delante y hacia atrás como desee con el directorio o saltar al primer o último elemento. El puntero de registro puede también ser cambiado usando el comando de búsqueda SEEK, que será descrito luego.

El estándar ISO/IEC 7816-4 también provee la opción de leer todos los registros desde el primero hasta un número de registro especificado. De forma similar desde un número de registro especificado hasta el último registro pueden ser leídos en un solo ciclo comando-respuesta usando el comando READ RECORD. Aunque estos comandos son muy prácticos, la capacidad del buffer E/S puede muy fácilmente ser excedida si ellos se usan con archivos grandes. La Figura 3.27 ilustra la ejecución de algunas operaciones de lectura y escritura sobre el archivo mostrado en la Figura 3.26.

nombre ↓	número telefónico ↓
Joshua Calvert	47 84 46
Quinn Dexter	099 11 00 11
Louise Kavanagh	099 47 51 22
Alkad Mzu	099 178 098
Iona Saldana	09933 178 234
Kiera Salter	099 23 76 87 33
Ralph Hiltch	099 78 123 78 1
Fletcher Christian	099 12 111 222
Kelly Tirrel	099 92 178 234
Gerald Skibbow	099 129 167 189

Figura 3.26 Ejemplo de una lista de números telefónicos en un archivo con una estructura 'lineal fija'. (*Referencia:* [WOLFGANG])

El comando APPEND RECORD, dada su funcionalidad, podría solo también ser clasificado como un comando de administración de archivos. Puede

ser usado para anexar registros a los archivos orientados a registros existentes. Los datos para el nuevo registro entero son provistos junto con el comando. Una relativamente compleja administración de memoria, en términos de TIs, es un prerrequisito para la disponibilidad de este comando con su funcionalidad completa. La función del administrador de memoria es crear un enlace entre el nuevo registro y los ya presentes en el archivo. Luego es posible, dentro de los límites de memoria disponible, añadir un número arbitrario de nuevos registros. Sin embargo, a menudo el número de nuevos registros que pueden ser añadidos es restringido para ahorrar recursos de memoria. En este caso, cuando es creado un archivo orientado a registros, se reserva memoria como necesaria para añadir futuros registros. Más tarde este espacio puede ser llenado usando comandos APPEND RECORD. Una vez que se gasta este espacio libre, el comando APPEND RECORD no puede ser usado otra vez en este archivo.

Smart card		Terminal
	←	READ RECORD *Comando* [código de regreso = "2"]
procesando el comando *Respuesta* [" Sheccid " || código de regreso]	→	IF (código de regreso = OK) THEN READ BINARY satisfactorio ELSE abortar
	←	UPDATE RECORD *Comando* [" Vera ", primero]
procesando el comando *Respuesta* [código de regreso]	→	IF (código de regreso = OK) THEN UPDATE BINARY satisfactorio ELSE abortar
	←	UPDATE RECORD *Comando* [" Patty ", siguiente]
procesando el comando *Respuesta* [código de regreso]	→	IF (código de regreso = OK) THEN UPDATE BINARY satisfactorio ELSE abortar
	←	READ RECORD *Comando* [número de registro = 2]
procesando el comando *Respuesta* [" Patty " || código de regreso]	→	IF (código de regreso = OK) THEN READ BINARY satisfactorio ELSE abortar

Figura 3.27 Operaciones simples de escritura lectura en un archivo orientado a registros. (*Referencia:* [WOLFGANG])

Si se usa APPEND RECORD junto con un archivo lineal fijo o variable, el nuevo registro siempre se añade al final del archivo. Si la estructura es cíclica, el nuevo registro siempre es numerado 1, lo cual corresponde al registro escrito en ese momento en archivos de ese tipo. APPEND RECORD puede ser usado para varios propósitos. Una posibilidad es un directorio telefónico, como ya fue mencionado. Otra posibilidad es un archivo de registro cronológico, en el cual los datos a ser registrados son escritos en la tarjeta directamente por la creación de nuevos registros.

3.3.3 COMANDOS SEARCH

Las estructuras orientadas a registros son bien adecuadas para almacenar conjuntos de datos relacionados con estructuras idénticas en un solo archivo. Un ejemplo típico es otra vez un directorio telefónico. Un comando SEARCH puede usarse para evitar tener que leer el directorio entero, registro por registro, cuando se busca un nombre particular.

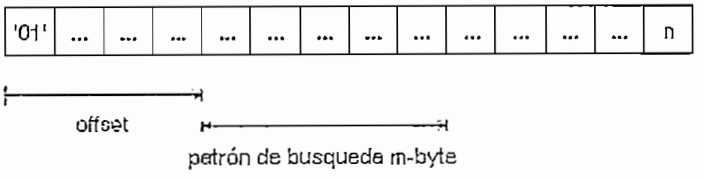

Figura 3.28 Buscando en un archivo orientado a registros. (*Referencia:* [WOLFGANG])

Tabla 3.9 Funcionalidad del comando SEEK acorde el estándar GSM 11.11. (*Referencia:* [WOLFGANG])

SEEK	
Comando	* longitud de la cadena buscada * cadena buscada * offset * modo de búsqueda (hacia delante desde el inicio, hacia atrás desde el final, hacia delante desde la siguiente posición, hacia atrás desde la posición previa) * switch: devuelve el número de registro del registro localizado
Respuesta	* número de registro (si es seleccionado por el switch) * código de retorno

El comando SEEK puede usarse para buscar una cadena de caracteres específica en una estructura de datos orientada a registros. Un offset puede ser suministrado con el comando. La longitud de la cadena buscada es variable. El comando debe decirle al SO en qué dirección buscar. Puede ser ya sea desde la posición de inicio hacia delante (en la dirección que el número de registro se incrementa) o desde la posición de inicio hace atrás. La posición de inicio para la búsqueda debe también ser especificada.

El primer registro, último registro o registro en uso puede ser especificado como la posición de inicio. Si la cadena buscada se encuentra, el SO pone el puntero de registro a la localización de la cadena e informa al terminal que la búsqueda ha sido satisfactoria.

Smart card		Terminal
	←	SEEK *Comando* [buscar cadena = " Sheccid " ‖ dirección de busqueda = " hacia delante desde el inicio" ‖ enviar número de registro]
Respuesta [número de registro = 6] [código de regreso]	→	IF (código de regreso = OK) THEN " Sheccid " encontrado ELSE " Sheccid " no encontrado
	←	SEEK *Comando* [buscar cadena = "Patty"‖ dirección de busqueda = " hacia atrás desde el final " ‖ enviar número de registro]
Respuesta [número de registro = 1] [código de regreso]	→	IF (return code = OK) THEN "Patty" encontrado ELSE "Patty" no encontrado

Figura 3.29 Secuencia simple de comando usando el comando SEEK (*Referencia:* [WOLFGANG])

El comando SEARCH BINARY puede usarse para buscar datos específicos en un archivo transparente seleccionado. El archivo podría ser seleccionado ya sea explícitamente por un comando transferido previamente o implícitamente vía un comando parámetro. El resultado es el offset del inicio del archivo al primer byte de la cadena de búsqueda localizada.

3.3.4 COMANDOS DE IDENTIFICACIÓN

Adicionalmente a usar TIs como un medio de almacenamiento de datos seguro, también se usan para identificación. El procedimiento usual involucra el intercambio de información secreta que es conocida solamente por el usuario y la tarjeta. Esto es usualmente un número de identificación personal, (PIN).

En la verificación del PIN, la tarjeta recibe en texto plano un código secreto (el PIN) vía la interfaz y solo tiene que compararse con el PIN mantenido en memoria. Interceptar la línea de transmisión de esta manera tendría consecuencias fatales. Para hacer imposible que tales ataques ocurran son diseñados procedimientos de autenticación modernos.

En principio, la autenticación involucra la verificación de un secreto conocido por ambas partes de la comunicación sin requerir que eso sea enviado a través del interfaz. Los procedimientos están construidos tal que la intercepción de los datos transmitidos no podría comprometer la seguridad de la autenticación. Dependiendo del SO, varios comandos están disponibles para la autenticación de la tarjeta o el terminal o ambos a la vez.

En muchos SOs, las llaves usadas para autenticación son protegidas por un contador de reintentos. Si el terminal intenta autenticaciones no satisfactorias un número determinado de veces, la tarjeta bloquea la llave asociada para evitar que se hagan pruebas de autenticación adicionales. Éste es un procedimiento perfectamente aceptable con respecto a la seguridad del sistema, pero eso tiene un inconveniente. El reseteo del contador de reintentos para la llave de autenticación a su valor inicial a menudo involucra muy complejos, difíciles logísticamente y costosos procedimientos administrativos. Consecuentemente, algunos sistemas no tienen contadores de reintentos para llaves de autenticación.

Por razones de seguridad, sólo llaves específicas para la tarjeta deberían usarse para autenticación. Esas llaves pueden ser generadas desde una única función de la tarjeta. Un número de serie o número de producción del chip es muy adecuado para este propósito. Tales números no confidenciales (y por tanto públicos) pueden ser leídos desde la tarjeta usando un comando adecuado. Actualmente no existen estándares al respecto; aquí usamos el comando GET CHIP NUMBER. El nombre del comando cambia de un SO a otro, como lo hacen los datos que son intercambiados.

El comando GET CHIP NUMBER obtiene un número de serie único de la tarjeta, el cual en vista del algoritmo DES debería preferiblemente ser de 8 bytes de longitud. Este número se usa únicamente para identificar el chip y calcular la llave especifica para la tarjeta.

Existe un comando más que es necesario para la autenticación. Éste es el comando GET CHALLENGE, el cual está especificado en ISO/IEC 7816-4. Se usa para solicitar un número generado al azar de la tarjeta. Este número subsiguientemente se usa durante la autenticación. Cuando se usa autenticación DES, la longitud del número es típicamente 8 bytes, pero éste podría ser diferente para otros algoritmos de criptografía.

El comando INTERNAL AUTHENTICATE permite al terminal autenticar una tarjeta o en el caso de una tarjeta multiaplicación, para autenticar una aplicación. Este comando puede ser usado para verificar que una tarjeta es genuina. La tarjeta recibe un número generado al azar, el cual es encriptado usando un algoritmo de criptografía como DES usando una llave que es conocida solo por la tarjeta y el terminal. El resultado de esta encripción es devuelto al terminal en la respuesta. El terminal realiza la misma encripción que la tarjeta y compara su resultado con el resultado en la respuesta recibida de la tarjeta. Si los dos resultados coinciden, se concluye que conoce la llave de autenticación secreta y debe ser genuina. La tarjeta ha sido de esta forma autenticada.

3.3.5 COMANDOS DE ADMINISTRACIÓN DE ARCHIVOS

La mayoría de COS modernos permiten varias operaciones de administración para ser realizadas sobre archivos dentro de límites puestos por condiciones de seguridad específica, incluyendo extensión, creación, borrado y bloqueo de archivos. No obstante, la mayoría o incluso todas las funciones de administración son frecuentemente omitidas en tarjetas de una sola aplicación, dado que esta galería de funciones requiere una gran cantidad de código de programa, lo cual podría incrementar la cantidad de memoria y por tanto el costo de la tarjeta. Con tarjetas multiaplicación, es esencial soportar ciertas funciones de administración para evitar tener que cargar todas las aplicaciones dentro de la tarjeta cuando es personalizada.

Con respecto a la seguridad del sistema, las funciones de administración deberían solo ser permitidas para ser ejecutadas luego de la autenticación mutua, dado que son puntos de inicio ideales para ataques.

Otro posible punto de ataque es concedido si las funciones de administración son usadas en terminales de acceso público, los cuales en principio son inseguros. En tales situaciones, la transferencia de datos siempre debe ser protegida usando funciones de intercambio de mensajes seguro. Solo luego un proveedor de aplicaciones podrá cargar seguramente archivos y aplicaciones en tarjetas ya en uso, por ejemplo vía tarjetas telefónicas públicas.

Particularmente en el caso de tarjetas multiaplicación, las cuales pueden usarse por algunos proveedores de aplicaciones, es necesario particionar la memoria y asignar llaves de autorización para la creación de archivos antes de que aplicaciones individuales sean generadas. Esto previene que cualquier proveedor de aplicación individual asigne la memoria entera disponible a su propia aplicación. Una forma de prevenir tal comportamiento es usar un

procedimiento que prelocalice espacios de memoria para cada aplicación y a la vez almacene llaves específicas de la tarjeta y de las aplicaciones para la creación de archivos en la tarjeta. Esto se puede hacer usando el comando REGISTER, el cual no está estandarizado. Nuevos archivos pueden ser creados más tarde si la llave de aplicación específica es conocida. Este método crea una separación estricta entre la localización de espacios de memoria y la carga de nuevos archivos en la tarjeta. El emisor de una tarjeta multiaplicación de esta manera puede vender espacio de memoria para algunos proveedores de aplicaciones sin tener que preocuparse por la piratería de memoria. El comando CREATE FILE permite que se creen DFs y EFs en TIs después de que éstas han sido completadas. Antes de que este comando pueda ser ejecutado, un estado lógico particular debe haber sido activado, por ejemplo mediante una autenticación mutua. Dependiendo del ambiente en el cual el comando CREATE FILE es ejecutado, la transferencia de datos podría haber sido protegida por intercambio de mensajes seguro. El comando CREATE FILE está definido en el estándar ISO/IEC 7816-9.

Después de que un archivo ha sido creado con todas sus condiciones acceso, atributos y otras propiedades, éste se puede seleccionar usando el comando SELECT FILE y luego accesado. El sistema operativo debe hacer imposible la creación de un archivo incompleto por la interrupción del proceso de creación de archivos, ya que un archivo semejante podría proveer una base para un ataque a la tarjeta. Además, no debe ser posible leer el contenido residual de áreas de memoria que están parcialmente sobrescritas cuando un nuevo archivo es creado.

3.3.6 COMANDOS PARA PROTOCOLOS DE TRANSMISIÓN DE DATOS

En principio, los protocolos de transmisión de datos deberían ser designados para ser completamente independiente de los datos y comandos de la capa aplicación. Desafortunadamente, existe una discrepancia aquí entre los

requerimientos teóricos y la práctica actual. Existen dos comandos cuyo único propósito es para habilitar que mecanismos de transporte serán utilizados al nivel de aplicación, estos son GET RESPONSE y ENVELOPE. Existe también otro comando, MANAGE CHANNEL, cuya función no se usa solamente por la capa aplicación.

En el protocolo T=0, no es posible enviar un bloque de datos a la TI y recibir un bloque de datos desde la TI con un solo ciclo comando-respuesta. Este protocolo por tanto no soporta comandos del caso 4[8], sin embargo ellos son usados frecuentemente. Por tanto es necesario usar un trabajo que rodee el protocolo T=0. Este opera de una manera simple. El comando del caso 4 primero es enviado a la tarjeta y si es satisfactorio, un código de regreso especial se envía al terminal para advertir al terminal que el comando ha generado datos que están esperando a ser recibidos. Luego el terminal envía un comando GET RESPONSE a la TI y recibe los datos en la respuesta. Esto completa el ciclo comando-respuesta del primer comando. Mientras no se envíe a la tarjeta otro comando que no sea GET RESPONSE, la respuesta de datos puede ser solicitada múltiples veces.

Si los comandos están completamente encriptados como parte de un intercambio de mensajes seguro, podrían ocurrir problemas de transmisión con el protocolo T=0, lo cual requiere tanto un byte instrucción de desencripción como un byte Le de desencripción. El comando ENVELOPE omite esta restricción por la implantación de una APDU completa, con su cabecera y sección de datos, en la sección de datos de la APDU del comando ENVELOPED. Éste puede ser encriptado sin ninguna restricción y transmitido usando cualquier protocolo. El mismo procedimiento se usa para la respuesta generada por la TI, la cual también es implantada en la APDU del comando ENVELOPED.

[8] Ver sección 3.2.7.1

Se pueden usar canales lógicos lo cual permite que cuatro aplicaciones en una sola TI puedan ser direccionadas independientemente una de otras. Los comandos y aplicaciones son coordinados por el uso de bits del byte clase. Antes de que el terminal pueda usar un canal lógico, éste debe explícitamente informárselo a la tarjeta mediante el comando MANAGE CHANNEL. Éste le dice a la tarjeta que un canal adicional es necesario. El número del canal puede ser especificado explícitamente por el terminal o la tarjeta puede suministrar el número de un canal libre en su respuesta. Cuando un nuevo canal lógico es abierto desde el canal estándar (canal 0), el comportamiento de la tarjeta en el nuevo canal es el mismo que después de un reset, lo cual significa que el MF es seleccionado y ningún estado de seguridad ha sido alcanzado. Cuando un nuevo canal lógico es abierto desde un canal que no sea el 0, la selección de DF y el estado de seguridad en uso en esos momentos son retenidos. Cuando un canal lógico es cerrado la selección de archivo asociada y el estado de seguridad son borrados.

Capítulo 4

Técnicas de seguridad

Una de las principales ventajas de las TIs en comparación con otros medios de almacenamiento de datos, tales como las tarjetas de banda magnética y disquetes, es que pueden almacenar datos que son protegidos y se mantienen en secreto. Un requerimiento esencial para esto es el hardware que se fabrica pensando en ello y está optimizado para este propósito, junto con los métodos de criptografía apropiados para la protección de datos confidenciales. Sin embargo, la seguridad depende no solo del hardware especial del microcontrolador y algoritmos implementados en el software del sistema operativo. La seguridad de las aplicaciones de las TIs y los principios de diseño usados por sus diseñadores, también son de fundamental importancia.

4.1 IDENTIFICACIÓN DEL USUARIO

Desde épocas remotas, una variedad de técnicas han sido usadas para la identificación diferenciada de personas. En el campo de tecnología de la información es una computadora la que debe identificar a un usuario en lugar de una persona como en el pasado. A pesar de su éxito al realizar actividades autómatas, las computadoras aun tienen tremenda dificultad en realizar tareas inteligentes. Consecuentemente, el ingresar una contraseña vía un teclado ha llegado a ser el método de identificación preferido. Los esfuerzos necesarios para la comparación son mínimos, dado que esencialmente todo lo que la computadora tiene que hacer es comparar la contraseña ingresada con un valor de referencia almacenado y hacer una simple decisión de si/no. La comparación de contraseñas eficazmente equivale a hacer una decisión referente a la autenticidad de la identidad de la persona que está siendo probada.

Básicamente existen sólo tres métodos que se pueden usar para identificar una persona. Si se usa una contraseña, lo que se prueba es que la persona conoce un secreto particular. Si el o ella lo conocen, la conclusión es que la persona sea el o ella quien dice ser. La segunda opción es para probar si una persona posee un objeto particular. La tercera posibilidad es para probar específicamente, características corporales únicas de una persona.

Los métodos que confían en el conocimiento de un secreto o en la posesión de un ítem tienen un inconveniente significante, el cual es que la persona que va a ser identificada también debe recordar algo de sí o llevar algo consigo. Dependiendo de la situación, el hecho de que el secreto pueda ser pasado a otra persona puede ser considerado como una ventaja o una desventaja. En cualquier caso, no es posible distinguir si la persona que conoce el código secreto o posee el ítem es verdaderamente su legítimo propietario, en lugar de alguien que ilegítimamente pudo haber adquirido el secreto o el ítem que está siendo probado.

El tercer método de identificación elimina esta transferabilidad, dado que se basa en características específicas del cuerpo humano para propósitos de identificación. Por supuesto, las mediciones son en la mayoría de los casos técnicamente difíciles, dado que por obvias razones las características biológicas que pueden ser medidas fácilmente, tales como el peso o la estatura no se pueden utilizar para este propósito.

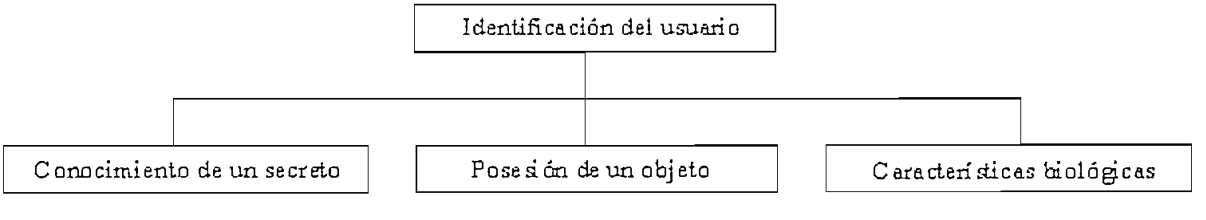

Figura 4.1 Clasificación de los métodos para la identificación de personas. (*Referencia:* [WOLFGANG])

Éste es el más fácil de entender de los tres métodos si consideramos el siguiente ejemplo. Suponga que tiene que encontrar a alguien desconocido en una estación del tren. Tan pronto como vea a un posible candidato, tiene el problema de decidir si es realmente la persona a la que está buscando. Sin embargo, si la persona desconocida aparece en el lugar indicado en el momento indicado, esto realmente equivale a una prueba implícita del secreto, dado que al menos se puede esperar que el lugar y momento de su encuentro sean generalmente desconocidos. Una prueba explicita de un secreto podría ocurrir si la persona desconocida pronunciara una contraseña que solo es conocida por Ud. y él o ella. Alternativamente, él podría identificarse por medio de un ítem que esté en su posesión, por ejemplo por llevar consigo un diario de un día específico bajo el brazo. Ciertamente, el método más seguro que podría ser chequeado a cerca de la persona es una característica corporal específica. Tal vez el o ella tiene una nariz muy larga, ojos azules, un peinado muy singular.

Este escenario de la estación de tren muestra claramente que la identificación de una persona desconocida puede ser considerada como un problema clásico que ocurre a diario y no se limita solamente a computadoras y tarjetas inteligentes.

Se ha convertido en una práctica común ingresar un PIN dentro de muchos tipos de equipos y computadores autómatas. El marcado incremento del número de PINs resultante para varios propósitos hace muy difícil que la gente común mantenga una pista de todos sus códigos PIN. Después de todo, ¿Quién puede recordar 20 o más PINs diferentes? La seguridad y buen nombre de un sistema naturalmente no son mejorados si cada usuario apunta su PIN sobre la tarjeta, dado que el número de casos de fraude podría ser excesivo. Por esta razón, la búsqueda de un método de identificación en lugar de los códigos PIN ha ido creciendo últimamente. Las características biométricas que permiten que una persona particular sea identificada sin ambigüedades por una máquina son ideales para este propósito.

4.1.1 IDENTIFICACIÓN DEL USUARIO USANDO UN PIN

El método que comúnmente es más usado para la identificación de usuarios es el ingreso de un número secreto, el cual generalmente es conocido como PIN (por 'personal identification number') o algunas veces CHV (cardholder verification).

Un PIN usualmente es un número de cuatro dígitos, compuesto de números decimales del 0 al 9. La razón de usar un ingreso puramente numérico es simplemente que los terminales de TIs generalmente sólo tienen un teclado numérico. El PIN se ingresa usando dicho teclado o un teclado de computador y luego se envía a la TI. La TI compara el valor que recibe con un valor de referencia almacenado internamente y comunica el resultado al terminal.

El ingreso del PIN es considerado particularmente como un tema de seguridad en aplicaciones de transacciones financieras, de modo que referencias relacionadas a la naturaleza del teclado se encuentran frecuentemente en esta área de aplicación. Los teclados especiales que satisfacen estos requerimientos a menudo se conocen como 'PIN pads '. En Alemania, por ejemplo, existe un requerimiento (de la ZKA) que el PIN para una tarjeta Eurocheque solo puede ser ingresado usando un teclado con protecciones mecánicas y criptográficas especiales. Los PIN pads tienen todas las características de un módulo de seguridad, como lo son sensores de abertura de cubiertas, láminas metálicas para evitar perforaciones y la encripción del PIN directamente al ser ingresado. Esto provee protección confiable contra la manipulación del teclado para permitir que el PIN sea interceptado mientras es ingresado.

Se puede hacer una distinción entre un PIN estático y uno modificable. Un PIN estático no puede ser cambiado por el usuario, por tanto efectivamente este debe ser memorizado por el usuario. Si éste llega a ser conocido, el usuario podría destruir la tarjeta y obtener una nueva con un diferente PIN. Un PIN

modificable puede ser alterado de acuerdo a los deseos del usuario o cambiado a un número que el usuario encuentre fácil de recordar. Existe un peligro en esto, dado que los números que muchas personas encuentran fáciles de recordar son por ejemplo: "1234", "4711" y "0815". La TI no hace un chequeo para el uso de tales números triviales, dado que no existe suficiente espacio de memoria para almacenar la tabla necesaria. Sin embargo, sería perfectamente concebible para el terminal, prohibir que el PIN sea cambiado a tales números. Para cambiar un PIN, siempre es necesario ingresar el PIN, dado que de otro modo el atacante podría siempre cambiar el PIN existente a uno de su propiedad.

La situación es diferente con las llaves personales de desbloqueo (PUKs 'personal unblocking keys'), las cuales también son conocidas como 'súper PINs'. Estas llaves usualmente tienen mas dígitos que un PIN normal (un valor típico es seis) y se utiliza para reiniciar el contador de reintentos de un PIN a cero si ha alcanzado su valor máximo. Un nuevo PIN también es ingresado dentro de la tarjeta cuando se ingresa el PUK, dado que el reiniciar el contador es poco útil si el usuario ha olvidado el PIN. Usualmente éste es el caso en el que el contador de reintentos ha alcanzado su máximo valor.

También existen aplicaciones que usan transporte de PINs. En este caso, la TI es personalizada con un PIN generado al azar y su propietario recibe el valor del PIN por correo. Luego el propietario reemplaza el PIN usado para la personalización de la tarjeta con un PIN de su elección antes de usar la tarjeta. En un método similar conocido como método 'null PIN', la tarjeta es precargada con un PIN trivial como por ejemplo '0000' y la TI obliga que el PIN sea cambiado antes de que la TI sea utilizada. Ambos métodos previenen que un PIN que ha sido 'espiado' durante la personalización de la tarjeta pueda ser utilizado luego para su uso indebido.

De acuerdo a la recomendación del estándar ISO 9564-1, el PIN debería consistir de cuatro a 12 caracteres alfanuméricos para minimizar la probabilidad

de determinar el PIN correcto por el método de prueba y error. Sin embargo, la situación en la práctica actual es a menudo algo diferente. El ingreso de caracteres alfanuméricos es técnicamente imposible en muchos lugares, dado que el teclado tiene solo caracteres numéricos.

El número de caracteres en un PIN depende no sólo de los niveles de seguridad deseados, sino también de un grado grande de capacidad de memoria del usuario de tarjetas promedio. Por mucho, la gente se ha acostumbrado a un PIN de cuatro dígitos, lo cual significa que cambiar a un PIN de seis dígitos o más podría ser muy difícil. En la práctica, la presunta mejora en seguridad provista por el uso de un PIN de 6 u 8 dígitos podría ser puramente teórica. Mucha gente encuentra difícil recordar números de esas longitudes, especialmente si aquellos no se usan muy a menudo y consecuentemente los escriben en la tarjeta o en un pedazo de papel que se mantiene cerca de la tarjeta. El nivel de seguridad con un PIN largo es entonces significativamente más bajo que con un PIN corto.

La perfectamente bien fundada insistencia sobre un periódico cambio de PIN tiene un destino similar. Puede surgir efecto con una aplicación de alta seguridad que tiene solo pocos usuarios, pero esto es fatal para la aceptación de una aplicación de mercado masivo, los cuales tratan de usar los métodos más simples posibles para acomodarse a la gente con memoria pobre.

4.1.1.1 Probabilidad de adivinar un PIN

El más simple ataque sobre un PIN, a parte de observar el PIN ingresado, es solo adivinarlo. La probabilidad de tener éxito depende en parte de la longitud del PIN, los caracteres de los cuales puede componerse y cuantos intentos están permitidos. La probabilidad de acertar un PIN de cuatro dígitos en tres intentos es 0,03 %, la cual no es particularmente alta. Dos formulas básicas para adivinar contraseñas están representadas aquí. Pueden ser usadas en la práctica actual para estimar el riesgo asociado al uso de una contraseña particular.

$$x = m^n \qquad (4.1)$$

$$P = i/m^n \qquad (4.2)$$

Donde:

i : número de intentos

m : número de posibles caracteres por posición

n : número de posiciones

P : probabilidad de adivinar el password

x : número de posibles passwords

Incidentalmente, hay aun un cuarto factor relacionado con adivinar el PIN, el cual por mucho tiempo ha estado inexcusablemente descuidado. Éste es la uniformidad de la distribución de PINs en una aplicación. Es más fácil adivinar un PIN si conoces que ciertos PINs son más comunes que otros. La importancia actual de este factor secundario llegó a ser evidente casi a finales de 1977 en conexión con las tarjetas German Eurocheque. Aunque el procedimiento detallado para el cómputo de un PIN de los datos almacenados en la banda magnética de la tarjeta Eurocheque todavía es secreto, al menos algunos pocos pasos generales del procedimiento son conocidos. De esta información, se podría concluir que los PINs que son generados no son uniformemente distribuidos, dado que el algoritmo usado produce números del 0 al 6 significantemente más a menudo que del 6 al 9. También llegó a ser conocido que los algoritmos de PIN suprimen los ceros principales cuando se generan PINs. Con algo semejante a una distribución no uniforme, no es necesario hacer 3333 intentos para adivinar correctamente un PIN de cuatro dígitos con el número permitido de intentos errados (3), sino solo 150 [WOLFGANG]. Con 10,5 % de las tarjetas, la distribución es tan pobre que solo 72 intentos serían suficientes para adivinar el PIN si las características del algoritmo de generación del PIN son tomadas en cuenta [WOLFGANG]. El resultado final de todo esto es que un algoritmo de generación de PIN mejorado está siendo utilizado en las tarjetas Eurocheque y que el algoritmo DES originalmente usado ha sido remplazado por el algoritmo triple DES.

4.1.1.2 Probando la autenticidad de un terminal

Como es bien sabido, ingresar un PIN es usado para verificar la identidad del usuario. Sin embargo, el usuario podría igualmente querer verificar que la identidad del terminal es genuina. El procedimiento que se sigue para evitar ataques involucra almacenar una contraseña en un archivo de la tarjeta. Esta contraseña es solo conocida por el usuario de la tarjeta y puede ser cambiada solo por él. Éste puede ser un número o un nombre elegido por el usuario. El COS permite el acceso de lectura a este archivo solo después de que el terminal ha sido autenticado por la tarjeta.

La primera cosa que pasa después de que el usuario ha insertado la tarjeta en el terminal es una transacción de autenticación mutua entre la tarjeta y el terminal. Si el resultado es exitoso, cada parte conoce que la otra parte es genuina. La tarjeta luego permite el acceso de lectura al archivo que contiene la contraseña secreta del usuario, la cual se muestra en el terminal. El usuario mira su contraseña y de este modo conoce que el terminal es genuino, toda vez que de otro modo el terminal no habría tenido acceso al contenido del archivo que guarda la contraseña secreta del usuario. El o ella pueden ahora ingresar con seguridad el PIN.

4.2 SEGURIDAD DE LA TARJETA INTELIGENTE

La característica esencial de una TI es que provee un ambiente de seguridad para los datos y programas. Si la cantidad de esfuerzos necesarios para leer de una TI no fuese tan grande, ésta podría ser esencialmente nada más que un simple disquete con una interfaz diferente.

Naturalmente, es prácticamente imposible configurar un sistema completo o incluso una TI, tal que tenga características de seguridad perfectas que hayan

sido probadas contra cualquier ataque. Si los esfuerzos puestos en los ataques están creciendo a niveles lo suficientemente altos, es posible conseguir el acceso a cualquier sistema y manipularlo. Sin embargo, cada atacante potencial hace un conciente o inconsciente análisis costo/beneficio para sí y sus objetivos. La recompensa de quebrantar un sistema debe valer el tiempo, el dinero y el esfuerzo invertidos en aquello que el atacante debe estar dispuesto a pagar para conseguir su objetivo. Independientemente de si la recompensa es dinero o prestigio dentro de un grupo con los mismos intereses, si no vale el esfuerzo, nadie invertirá muchas energías en quebrantar un sistema o una TI.

La seguridad de una tarjeta está garantizada por cuatro componentes. El primer componente es el cuerpo de la tarjeta, en el cual está incrustado el microcontrolador. Muchas de las características de seguridad usadas en el cuerpo de la tarjeta no sólo pueden ser leídas por una máquina, sino también pueden ser visualizadas por los humanos. Las técnicas usadas para esas características no son específicamente para TIs, también se usan en otros tipos de tarjetas. Los componentes restantes –el hardware del chip, el COS y la aplicación – protegen los datos y programas en el microcontrolador de la TI.

Figura 4.2 Clasificación de los componentes de seguridad de las Tarjetas Inteligentes. (*Referencia:* [WOLFGANG])

La seguridad de una TI está garantizada sólo cuando todos estos componentes están presentes y sus mecanismos de defensa están trabajando apropiadamente. Si se utiliza la tarjeta exclusivamente en un ambiente donde no está sujeta a la verificación humana, los componentes del cuerpo de la tarjeta no son necesarios. Los tres componentes que son independientes del cuerpo de la

tarjeta, sin embargo, son indispensables para la seguridad física y lógica de una TI con respecto a ataques. Si alguno de estos componentes falla o si alguno no conoce los requerimientos aplicables, la TIs ya no está segura por más tiempo, toda vez que estos componentes mantienen una relación AND lógico entre sí.

El tiempo de vida útil de una TI es generalmente tres años. El reto de los fabricantes de microcontroladores de TIs y productores de COSs es mantener la delantera sobre todos los atacantes. Esto permite en consecuencia que los posibles atacantes sean minimizados o evitados por el empleo adecuado de contramedidas.

4.2.1 CLASIFICACIÓN DE ATAQUES Y ATACANTES

El problema principal enfrentado por todos los sistemas de tecnología de la información que son sujeto de ataques es el 'efecto avalancha', lo cual se observa a menudo después de que un ataque es realizado. El efecto avalancha consiste en que si un atacante encuentra una vulnerabilidad en algún sistema no pasará mucho tiempo antes de que esta información se comunique al mundo entero vía Internet donde incluso se podría ofertar software, hardware e incluso documentación referente a dicha vulnerabilidad y que de hecho facilitará que muchos puedan aprovecharse de esto.

A continuación se intenta clasificar los diferentes tipos de ataques y atacantes. El énfasis naturalmente consiste en los aspectos de tecnologías de la información de las TIs, antes que las características de la seguridad del cuerpo de la tarjeta que pueden ser chequeadas por humanos. Esta clasificación permite a los ataques potenciales ser evaluados a fin que se pueden tomar medidas sobre ellos. Como es bien sabido, es más fácil defenderse contra ataques conocidos que contra ataques desconocidos.

Hemos basado la clasificación de los diferentes tipos de ataques en el estándar ISO 13491-1, el cual describe los conceptos de requerimientos y métodos de evaluación para el equipamiento de seguridad criptográfica en el sector bancario.

4.2.1.1 Clasificación de ataques

La intención es presentar un resumen general de métodos de ataque y defensa que no es específica para un sistema en particular.

En principio, los ataques sobre las TIs pueden dividirse en tres tipos diferentes: ataques al nivel social, ataques al nivel físico y ataques al nivel lógico. Naturalmente, los tipos de ataques mixtos también ocurren en la práctica. Por ejemplo, un ataque al nivel físico podría preparar la vía para un ataque subsiguiente al nivel lógico.

Los ataques al nivel social son ataques que principalmente están dirigidos contra las personas que trabajan con TIs. Éstos pueden ser diseñadores del chip trabajando para fabricantes de semiconductores, diseñadores de software o, más adelante en el ciclo de la vida de la TI, el portador de la tarjeta. Estos ataques solo pueden ser parcialmente contrarestados por las medidas técnicas. Deben ser contrarestados en primer lugar por las medidas organizacionales. Adquirir disimuladamente un PIN mirándolo mientras es tecleado puede ser impedido fácilmente suministrando pantallas visuales sobre cada lado del teclado. Los ataques al nivel social en contra de los programadores de tarjetas se vuelven inútiles por la construcción de los procedimientos públicos usados. En este caso, la seguridad depende solo de las llaves secretas y de los conocimientos que tenga el desarrollador de software no le sirven a un atacante. Los ataques sobre las TIs al nivel físico usualmente requieren equipo técnico, dado que es necesario obtener acceso físico al hardware del microcontrolador de la TI en una forma u otra. Tales ataques pueden ser ya sea estáticos, lo cual significa que no es

necesario un suministro de energía aplicado al microcontrolador o dinámicos, con el microcontrolador funcionando. Los ataques físicos estáticos no imponen restricciones de tiempo sobre el atacante, quien puede hacer su trabajo con su propio ritmo. Con un ataque dinámico, por contraste, el atacante debe tener equipos de acceso rápido para la adquisición y evaluación de los datos.

Hasta ahora, la mayoría de ataques exitosos sobre TIs han sido a nivel lógico. Esos ataques crecen de la pura reflexión o cálculo mental. Esta categoría incluye criptoanálisis clásico, así como ataques que explotan los defectos conocidos de los COS y caballos de Troya en el código ejecutable de las aplicaciones de TIs.

Solo con el criptoanálisis de los protocolos de criptografía, esos ataques pueden ser divididos en tipos pasivos y activos. En un ataque pasivo, el atacante analiza el texto cifrado o protocolo de criptografía sin modificarlo y podría por ejemplo hacerlo midiendo el dispositivo semiconductor. En un ataque activo, por contraste, el atacante manipula el proceso de transmisión de datos o el microcontrolador.

Si se observa el ciclo de vida de una TI, podemos hablar de posibles ataques en tres fases que son: (a) desarrollo, (b) producción y (c) uso de la tarjeta.

Los ataques durante el desarrollo tienen relación con el diseño del sistema, desarrollo del chip, desarrollo del sistema operativo y la generación de aplicaciones, El término 'producción' se usa en este contexto para referirse en general a todo proceso utilizado para hacer hardware. Este cubre la totalidad del rango desde la fabricación de la hostia por los fabricantes de semiconductores hasta la personalización y envío de la tarjeta al usuario. El uso de la tarjeta se refiere a la fase en la cual las TIs están en el campo, lo cual significa cuando han sido usadas por el portador.

4.2.1.2 Clasificación de atacantes

Todos los atacantes pueden ser igualmente peligrosos para un sistema de TIs, pero tienen diferentes capacidades y opciones. Un hacker típico, por ejemplo, tiene una cantidad moderada de conocimientos del sistema, buenas ideas creativas usualmente un similar grupo de amigos. Él normalmente no posee una gran cantidad de equipo y sus finazas son limitadas. Sin embargo, si él es competente y utiliza el método adecuado, ciertamente podrá obtener acceso a una gran cantidad de capacidades de procesamiento, por ejemplo mediante una campaña en Internet.

Todos los 'insiders' forman una clase especial de atacantes, bajo la suposición de que ellos tienen buen conocimiento del sistema. Ellos podrían tener acceso a los componentes de software y hardware y podrían ser concientes de las debilidades del sistema. Mientras sean individuos sencillos los involucrados, pueden ser considerados como 'hackers' en términos de sus recursos y opciones. Sin embargo, dado que los 'insiders' no son ni anónimos ni espacialmente numerosos, usualmente es posible identificar la fuente de sus ataques.

La tercera clase de personas que pueden ser consideradas como atacantes potenciales son los criminales o 'criminals'. A pesar de que ellos usualmente no tienen un alto nivel de conocimientos técnicos, exhiben considerable energía cuando de obtener beneficios personales se trata (principalmente financieros) como resultado de sus actividades.

Una fuente potencial de ataques que no puede ser ignorada en la práctica consiste de las instituciones académicas, tales como universidades e institutos técnicos, incluyendo sus estudiantes y profesores. Ellos no necesariamente tienen conocimientos especiales de microcontroladores o aplicaciones de TIs en particular, pero poseen grandes cantidades de conocimientos generalmente útiles. Adicionalmente, ellos tienen acceso a una gran cantidad de recursos calificados y

no caros en forma de estudiantes y graduados, así como del equipo adecuado en sus laboratorios. Muchas de esas instituciones también albergan una inmensa cantidad de capacidad de procesamiento y gente altamente motivada con una tendencia experimental.

Una clase especial de atacantes está formada por los competidores o 'competitors'. Ellos normalmente tienen considerables conocimientos técnicos y algunos de ellos podrían tener equipo de análisis muy sofisticado.

Las organizaciones de crimen organizado naturalmente representan un nivel completamente diferente de atacantes de TIs. Tiene suficiente financiamiento para adquirir todos los conocimientos y herramientas necesarias para un ataque exitoso, ya sea comercialmente o por medios ilícitos.

4.2.1.3 Clasificación de la atracción de un ataque

Para permitir que medidas de un perímetro de defensa sean puestas en un lugar, las atracciones de un ataque deberían ser evaluadas para cada debilidad potencial del sistema. Esto se puede hacer de una manera matemática objetiva usando análisis de valor, para calcular una lista de prioridad de los probables blancos de ataque. El esquema que se presenta a continuación está simplificado, pero aun nos permite hacer una estimación relativamente buena de los atractivos de varios tipos de ataques y por tanto de las probables líneas de ataque. Naturalmente, un atacante podría normalmente escoger un ataque que requiera el menor precio y esfuerzo. Los seis criterios listados en la Tabla 4.1 influenciarán conciente o inconscientemente el comportamiento del atacante.

El nivel más bajo de conocimientos específicos o habilidades requeridos por un atacante, es el más atractivo para un individuo u organización. De igual manera, un ataque que no requiere el conocimiento de algún secreto es más

atractivo que alguno que requiera el conocimiento de varios secretos. Esto no es inconsistente con el principio de Kerckhoff, el cual dice que la seguridad dependerá solamente de la llave y no del algoritmo de criptografía como tal, toda vez que el principio de Kerckhoff no significa que sea necesario revelar todo acerca de un sistema para hacerlo seguro. La presencia de muchos secretos representa un enorme obstáculo para montar un ataque exitoso.

Tabla 4.1 Criterio para determinar el esfuerzo y costo requerido para un ataque sobre los componentes de seguridad de hardware o software, basado en los prerrequisitos para un ataque. (*Referencia:* [WOLFGANG])

Grado de atracción	Bajo	Medio	Alto
Nivel de conocimientos y habilidades	Alto	Medio	Bajo
Número de secretos requeridos	Muchos	Moderado	Pocos
Cantidad de tiempo requerido	Mucho	Moderado	Corto
Adquisición del equipo técnico necesario (compra o acceso)	Difícil	Moderado	Fácil
Acceso a los objetivos de ataque	Difícil	Moderado	Fácil
Valor de los resultados	Bajo	Medio	Alto

Especialmente en la búsqueda sistemática de una llave, la cantidad de tiempo requerido juega un papel muy importante. Los atractivos de un ataque igualmente son dependientes del equipo requerido para el ataque. Lo cual no solo se refiere a la compra del equipo, toda vez que éste podría estar disponible para rentarlo o alguien podría tener acceso a él. La disponibilidad de los componentes a ser atacados influencia fuertemente el atractivo de un particular tipo de ataque. Un atacante puede trabajar tranquilamente en su propia habitación, estudiando las comunicaciones y comportamiento de su propia tarjeta para tratar de duplicarla usando una computadora, sin ser observado por nadie más y sin interferencia en su trabajo. Sin embargo, si él hiciera lo mismo con un terminal en un supermercado, el cajero inmediatamente podría prohibir cualquier experimento e interrumpir el trabajo.

El criterio final, el cual es de decisiva importancia, es naturalmente el valor del resultado de los esfuerzos del atacante. Sus esfuerzos deben ser recompensados, ya sea de forma monetaria o acrecentando su prestigio.

4.3 CRIPTOGRAFÍA

En el campo de las Tarjetas Inteligentes, el uso práctico de los procedimientos y métodos criptográficos existentes representan la tarea principal y área de aplicación primaria con respecto a la criptografía. Consecuentemente, aquí nos concentraremos más en los aspectos prácticos de la criptografía que en los aspectos teóricos.

Los cuatro objetivos de la criptografía son mantener los mensajes secretos en secreto *(confidencialidad)*, asegurar la *integridad* y la *autenticidad* de los mensajes y asegurar la capacidad de validar los mensajes *(no repudiación)*. Confidencialidad significa que sólo el receptor predeterminado de un mensaje puede deencriptar su contenido. Autenticidad significa que el receptor puede verificar que el mensaje recibido no haya sido alterado en el curso de su transmisión. No repudiación significa que el remitente puede verificar que un cierto destinatario ha recibido un mensaje particular, lo cual significa que el mensaje tiene la capacidad de ser validado.

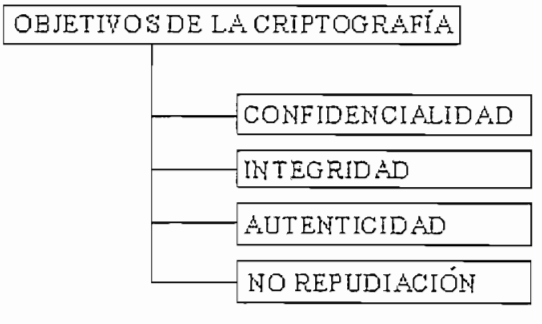

Figura 4.3 Clasificación de los cuatro objetivos independientes de la criptografía. (*Referencia:* [WOFLGANG])

En términos simplificados, existen tres tipos de datos en la tecnología de encripción. El primero es *texto plano o plaintext,* el cual es un dato no encriptado, Cuando un dato está encriptado su denominación es *texto cifrado o ciphertext.* Finalmente existe una llave, una o más de las cuales se requieren para la

encripción y deencripción. Esos tres tipos de datos son procesados por un algoritmo de encripción. Los algoritmos que actualmente se utilizan en TIs generalmente son orientados al bloque, lo cual significa que el texto plano y texto cifrado solo pueden ser procesados en paquetes con longitudes fijas (como 8 bytes con DES).

Los algoritmos de criptografía se dividen en dos tipos: simétricos y asimétricos. Esta división está basada en la llave que se utiliza. Aquí 'simétrico' significa que el algoritmo usa la misma llave para la encripción y deencripción. En contraste, los algoritmos asimétricos (los cuales fueron postulados en 1976 por Whitfield Diffie y Martin E. Hellman) usan diferentes llaves para la encripción y deencripción.

Un término que a menudo aparece con los algoritmos de criptografía es la magnitud del espacio de la llave. Esto se refiere al número de posibles llaves que pueden ser usadas por un algoritmo de criptografía en particular.

Un requerimiento que sólo recientemente ha llegado a ser prominente con respecto a la implementación técnica de los algoritmos de criptografía en una TI es la libertad de ruido. En este contexto, esto significa que el tiempo de ejecución de un algoritmo no debe depender de la llave, del texto plano ni del texto cifrado. Si este requerimiento no es satisfecho, sería posible descubrir la llave en un tiempo relativamente corto, lo cual podría significar que el sistema de criptografía completo fuese roto.

Un sistema criptográfico puede asegurar la confidencialidad y/o autenticidad de un mensaje. Si el sistema ha sido roto, esto significa que la confiabilidad y/o autenticidad ya no están garantizadas por más tiempo.

Algunos métodos de ataques diferentes pueden usarse para romper la llave de un algoritmo de criptografía. En un ataque 'solo de texto cifrado', el atacante conoce solo el texto cifrado e intenta determinar la llave o texto plano a partir del texto cifrado. Un método mas prometedor de ataque es el de 'texto plano conocido', el cual involucra el conocimiento por parte del asaltante de varios pares texto plano-texto cifrado para una llave secreta. Los ataques de 'texto plano seleccionado' y 'texto cifrado seleccionado' requieren que el atacante esté habilitado para generar sus propias parejas texto plano-texto cifrado. Si esto es posible, la probabilidad de éxito es mejorada, dado que la llave secreta puede ser descubierta experimentalmente.

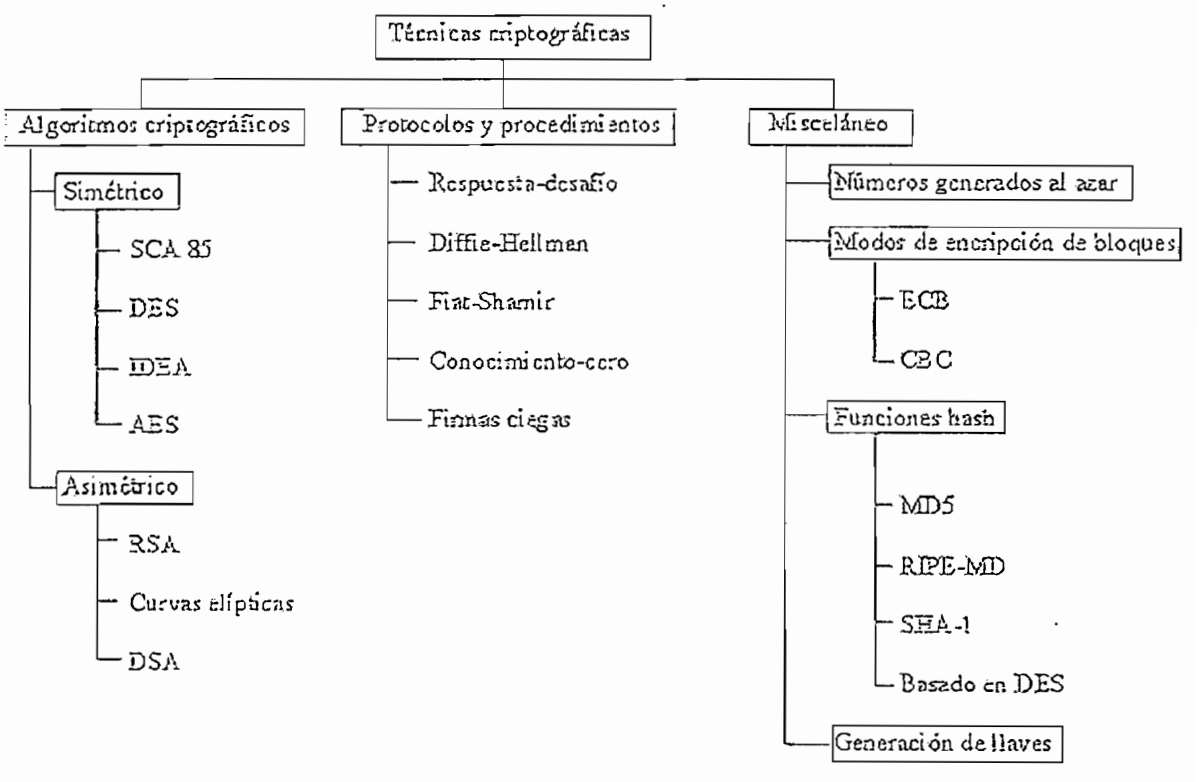

Figura 4.4 Diagrama de la clasificación de las técnicas criptográficas usadas en el área de las Tarjetas Inteligentes. (*Referencia:* [WOLFGANG])

Descubrir una llave por el método de intento-error es el método de ataque menos sofisticado. Con este método, se hace un intento para enterarse de la llave correcta empleando una gran cantidad de capacidad de procesamiento para

probar todas las llaves posibles con el conocimiento previo de una pareja de texto plano y texto cifrado.

4.3.1 ALGORITMOS CRIPTOGRÁFICOS SIMÉTRICOS

Los algoritmos criptográficos simétricos están basados en el principio de realizar la encripción y deencripción usando la misma llave secreta – de allí la designación de 'simétrico'.

4.3.1.1 El Algoritmo DES

El algoritmo simétrico más conocido y usado es el Data Encryption Algoritm (DEA), el estándar que describe el DEA generalmente se lo llama DES (Data Encryption Standard). Dos importantes principios para un buen algoritmo de encripción fueron incorporados en el diseño del algoritmo DES. Esos principios son confusión y difusión, propuesto por C. Shannon en un inicio. El principio de confusión establece que las estadísticas del texto cifrado deberían afectar las estadísticas del texto plano de manera que sea muy complejo que un atacante pudiera tomar alguna ventaja de él. El principio de difusión establece que cada bit del texto plano y cada bit de la llave deberían afectar tantos bits como sea posible del texto cifrado.

DES, el cual es un algoritmo de encripción simétrico de bloque, no aumenta el tamaño del texto cifrado. Esto significa que los bloques de texto plano y texto cifrado tienen el mismo tamaño. El tamaño del bloque es de 64 bits (8 bytes), el cual también es la longitud de la llave, sin embargo actualmente solo 56 de esos bits se usan para la llave. La llave contiene ocho bits de paridad, lo cual reduce el espacio disponible para la llave. Los 64 bits de la llave se numeran consecutivamente de izquierda (msb) a derecha (lsb). Los bits 8, 16, 24, ..., 64 son los bits de paridad. La paridad siempre es impar. Debido a los bits de paridad,

el espacio de la llave es 2^{56}, lo cual significa que existen aproximadamente 7.2 x 10^{16} llaves posibles. A primera vista, un espacio de llaves con 72.057.594.037.927.936 posibles llaves podría parecer muy grande, pero el tamaño limitado de su espacio de llaves es la principal debilidad del algoritmo DES. Dada la capacidad de procesamiento firmemente creciente de las computadoras modernas, un espacio de llave de este tamaño ya es considerado como pequeño para un algoritmo de criptografía. Si se tiene disponible un par de texto plano-texto cifrado, es muy fácil probar todas las llaves posibles en este espacio tan pequeño. Consecuentemente, el algoritmo triple-DES se usa casi exclusivamente para aplicaciones nuevas.

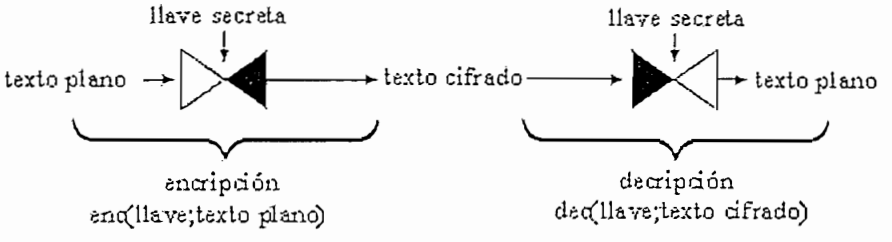

Figura 4.5 Principio de operación de operación de un algoritmo simétrico. (*Referencia:* [WOLFGANG])

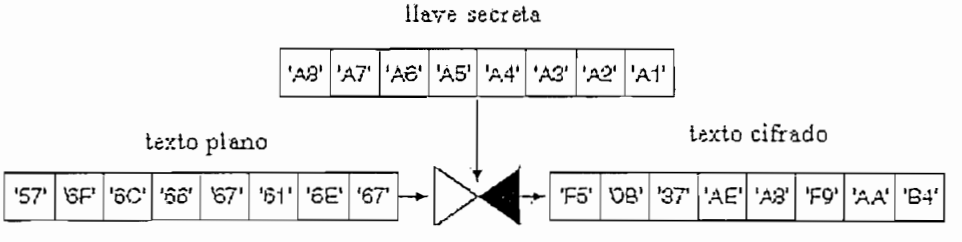

Figura 4.6 Operación del algoritmo DES para encriptar datos. (*Referencia:* [WOLFGANG])

El DES fue diseñado como un algoritmo de encripción que puede ser implementado fácilmente en hardware. Actualmente existen muchos microcontroladores de TIs con un módulo de hardware DES. Sin embargo, si fuera necesario implementar DES en una TI en software, esto ocuparía alrededor de 1 kb de código en assembler, aun cuando sea una versión optimizada.

Los típicos tiempos de cómputo para la encripción y deencripción usando una TI, en comparación con valores al usar una PC y un circuito integrado en hardware, se muestran en la Tabla 4.2. Esos números pueden variar de acuerdo a la implementación actual y toman en cuenta solo los tiempos de procesamiento puramente para encripción o deencripción DES de un bloque de 8 bytes, asumiendo que todos los registros ya están cargados. Como regla general, se puede asumir que una implementación de DES en hardware es aproximadamente 150 veces mas rápida que DES en software.

Tabla 4.2 Tiempos de cómputo típicos para DES. (*Referencia:* [WOLFGANG])

Implementación	Tiempos de cómputo / throughput
TI, reloj de 3.5-MHz, implementación en software	17,0 ms / 3,8 kbit/s
TI, reloj de 3.5-MHz con unidad de procesamiento DES	112 µs / 571 kbit/s
TI, reloj de 3.5-MHz con unidad de procesamiento 3 DES	130 µs / 492 kbit/s
TI, reloj de 4.9-MHz, implementación en software	12,0 ms / 5,3 kbit/s
TI, reloj de 4.9-MHz con unidad de procesamiento DES	80 µs / 800 kbit/s
TI, reloj de 4.9-MHz con unidad de procesamiento 3 DES	93 µs / 688 kbit/s
PC (80486, 33Hz)	30 µs / 2,1 MB/s
PC (Pentium, 90 MHz)	16 µs / 4 MB/s
PC (Pentium, 200 MHz)	4 µs / 16 MB/s
Circuito integrado DES	64 ns / 100 MB/s

Las llaves para el algoritmo DES pueden ser creadas usando un generador de números ramdómicos que produce un número ramdómico de 8 bytes, el cual luego es chequeado contra las cuatro llaves débiles y las 12 llaves semi-débiles. Si los valores calculados no coinciden con alguna de estas llaves fácilmente quebrantables, se calculan los bits de paridad y el resultado es una llave DES.

4.3.1.2 El Algoritmo IDEA

Existen muchos otros algoritmos simétricos de criptografía junto con DES, uno de ellos es el Algoritmo Internacional de Encripción de Datos (IDEA). El cual al igual que DES, es un algoritmo de encripción orientado al bloque, la longitud de la llave es de 16 bytes. Esto provee un espacio de llaves significativamente mas grande, con un tamaño de $2^{128} \approx 3.4x10^{38}$. En notación decimal regular, el número

de posibles llaves para el algoritmo IDEA es exactamente 340.282.366.920.938.463.463.374.607.431.768.211.456.

Existen solo pocas TIs con IDEA implementado en ellas. La cantidad de espacio en memoria necesario para el programa es alrededor de 1000 bytes. Típicamente los tiempos de cómputo para encripción y deencripción son algo menos que para DES. Sin embargo, en el desarrollo de IDEA, se asumió que los cálculos serían realizados por un procesador de 16 bits. Dado que la mayoría de TIs aun tienen procesadores de 8 bits, la ventaja de velocidad con respecto a DES no es tan grande como podría esperarse. La tabla 4.3 lista valores típicos para las operaciones IDEA sobre un bloque de 8 bytes, asumiendo que llaves previamente calculadas están disponibles.

Tabla 4.3 Tiempos de cómputo típicos para IDEA. (*Referencia:* [WOLFGANG])

Implementación	Tiempos de cómputo
TI, reloj de 3.5-MHz, implementación en software	12,3 ms
TI, reloj de 4.9-MHz, implementación en software	8,8 ms
PC (80486, 33Hz)	70 µs
PC (Pentium Pro, 180 MHz)	4 µs
Circuito integrado IDEA	370 ns

4.3.1.3 Modos de operación de los algoritmos de encripción orientados a bloques

El algoritmo DES, como todo algoritmo de encripción orientado al bloque, puede ser usado en cuatro diferentes modos de operación que están estandarizados en ISO 8372. Dos de esos modos de operación, los modos CFB y OFB, son adecuados especialmente para texto secuencial que no son estructuras de bloque. Los otros dos, los modos ECB y CBC, se basan en tamaños de bloque de 8 bytes. Esos dos modos de operación orientados al bloque son muy usados en aplicaciones de TIs.

El modo básico de operación del algoritmo DES está diseñado como el modo ECB (electronic code book). En este modo, los bloques de 8 bytes de texto plano son encriptados independientemente usando una sola llave. Éste es el algoritmo DES en su forma pura, sin extensiones.

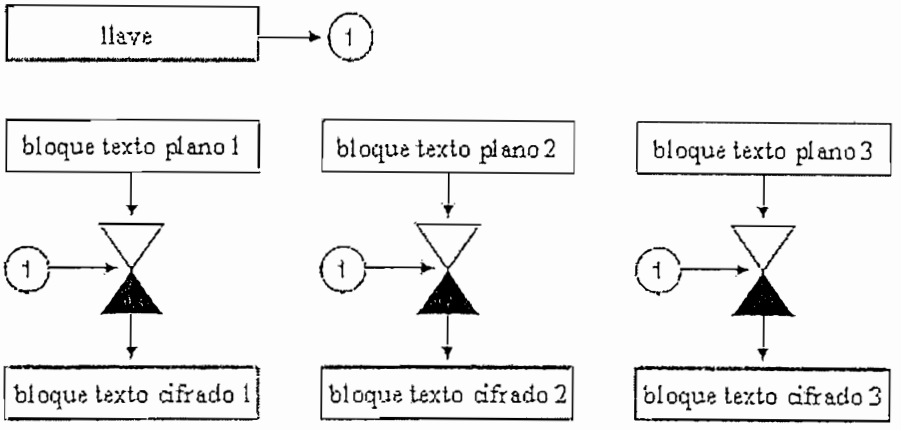

Figura 4.7 Encripción de datos con un algoritmo de encripción orientado al bloque en el modo ECB. (*Referencia:* [WOLFGANG])

El segundo modo de operación orientado al bloque es conocido como modo CBC (cipher block chaining). En este modo, una cadena de datos que consiste de algunos bloques es encadenada usando una operación XOR durante la encripción tal que cada bloque llega a ser dependiente del bloque que lo precede. Esto hace posible que una fuente fidedigna detecte algún intercambio, añadidura o borrado de bloques encriptados. Esto no es posible en el modo ECB. Si los bloques de texto plano están adecuadamente estructurados (con contadores de secuencia en sus cabeceras o vectores de inicialización), una consecuencia del encadenamiento CBC es que incluso bloques de texto plano idénticos son convertidos en bloques de texto cifrado no idénticos. Esto hace el criptoanálisis de datos interceptados mucho más difícil.

El primer bloque texto plano pasa por una operación XOR con un vector de inicialización (a menudo conocido como IV) y luego es encriptado con el algoritmo DES. El resultado es el texto cifrado, el cual a su vez pasa por una operación XOR con el siguiente bloque de texto plano. El proceso continúa de esta manera

con los siguientes bloques. Como una regla, el vector de inicialización esta preconfigurado como nulo. Sin embargo, en algunos sistemas, un número ramdómico de sesión específica es escrito en el vector de inicialización como un sustituto de una llave temporal. Este número naturalmente debe ser conocido cuando los datos son subsecuentemente deencriptados.

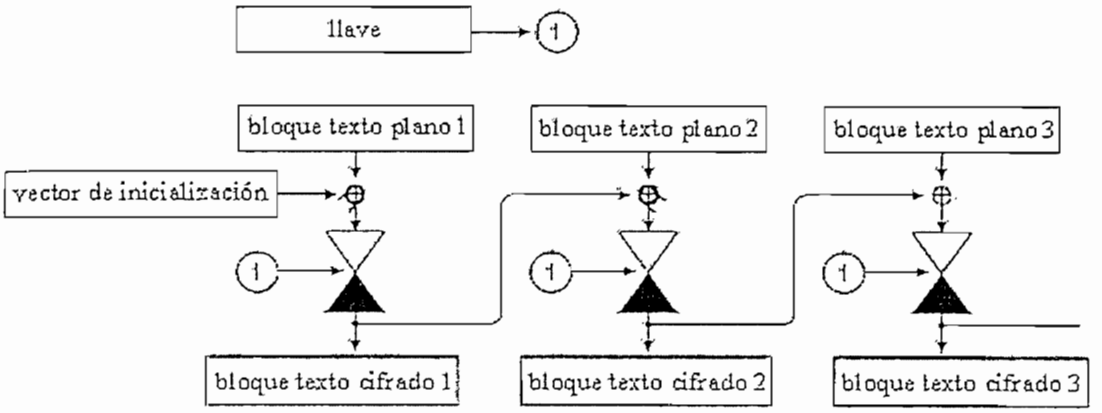

Figura 4.8 Encripción de datos con un algoritmo de encripción orientado al bloque en el modo CBC. (*Referencia:* [WOLFGANG])

4.3.1.4 Encripción múltiple

El proceso más utilizado de este tipo es el llamado triple-DES. En este modo, se realizan tres operaciones DES en modo CBC usando alternadamente encripción y deencripción. Los bloques que han sido encriptados de esta manera son deencriptados en el orden inverso de operaciones (en otras palabras: deencripción, encripción y luego deencripción). Si las tres llaves son la misma, el resultado de las operaciones alternada de encripción y deencripción es el mismo que se obtendría por una sola encripción. Esta es la razón por la que no se usa una secuencia de tres operaciones de encripción.

Si las tres operaciones de DES usando tres llaves se aplican directamente a cada bloque de texto plano en turno, el proceso se conoce como 'DES en el interior del modo CBC'. Si en lugar de eso el texto plano es completamente encriptado usando la primera llave y el resultado luego es encriptado de una

manera similar, el proceso se conoce como 'DES en el exterior del modo CBC'. El segundo es el modo más resistente a ataques y por consiguiente es generalmente recomendado.

El algoritmo triple DES tiene algunos nombres, tales como TDES, DES-3, 3-DES y 2-DES. Actualmente, los términos 'triple DES' y '3-DES' solo significan que tres llaves de 56 bytes de longitud son usadas. Si la primera y la tercera llaves son la misma, éste es conocido como 2-DES, pero si las tres llaves son todas diferentes, la designación que a menudo se usa es 3-DES. La longitud de llave siempre debe ser establecida con triple DES para que ninguna ambigüedad se presente en el algoritmo.

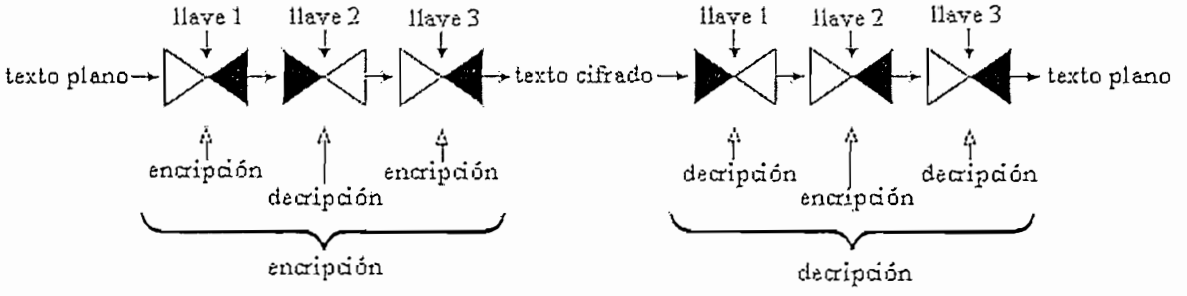

Figura 4.9 Principio de operación de encripción de datos usando triple DES. (*Referencia:* [WOLFGANG])

4.3.2 ALGORITMOS CRIPTOGRÁFICOS ASIMÉTRICOS

En 1976, Whitfield Diffie y Martin E. Hellman describieron la idea de desarrollar un algoritmo de encripción basado en dos llaves diferentes. Una de esas llaves fue la llave pública, la otra secreta o 'privada'. Esto podría permitir encriptar un mensaje usando la llave pública, con lo que solo el propietario de la llave privada podría deencriptarlo. Esto con el fin de eliminar los problemas asociados con el intercambio y distribución de llaves secretas simétricas. Adicionalmente, por primera vez podría ser posible realizar otros procesos, tales como generar firmas digitales que podrían ser generadas por cualquiera.

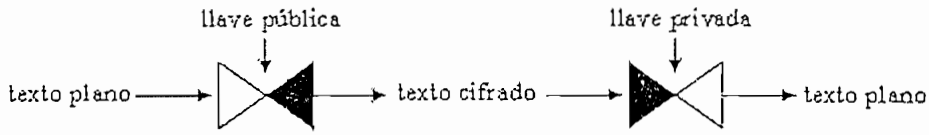

Figura 4.10 Encripción y deencripción usando un algoritmo de llave pública. (*Referencia:* [WOLFGANG])

4.3.2.1 El Algoritmo RSA

Dos años más tarde, Ronald L. Rivest, Adi Shamir presentan un algoritmo que se basa en el criterio antes mencionado. Este algoritmo, el cual es conocido como algoritmo RSA por las iniciales de sus inventores, es el algoritmo asimétrico de criptografía más conocido y versátil que está en uso en la actualidad. Su principio de operación muy simple se basa en la aritmética de enteros grandes. Las dos llaves se generan de dos números primos grandes.

El proceso de encripción y deencripción puede ser expresado matemáticamente como sigue:

encripción: $y = x^e \bmod n$

deencripción: $x = y^d \bmod n$

donde

x = texto plano

y = texto cifrado

e = llave pública

d = llave privada

n = módulos públicos = p.q

p,q = números primos secretos

4.4 ATAQUES Y CONTRAMEDIDAS DURANTE EL DESARROLLO.

Ya en la fase de desarrollo del hardware del microcontrolador y el software del sistema operativo de la TI, una variedad de medidas de seguridad son tomadas. Similar a la calidad de un producto la seguridad debe considerarse desde el propio inicio del desarrollo de un producto porque no puede ser añadida al diseño de un producto en un punto posterior de tiempo.

4.4.1 DESARROLLO DEL MICROCONTROLADOR DE LA TI.

El desarrollo del hardware de un microcontrolador de la TI toma muchos meses y es llevado efectuado por pocas personas de un fabricante de semiconductores en instalaciones seguras y monitoreadas. Los sistemas de cómputo correspondientes para el diseño del semiconductor están típicamente conectados a una red independiente que no tiene punto de contacto con el resto del mundo. Esto garantiza que fuentes externas no pueden implementar modificaciones al diseño del chip ni determinar la arquitectura interna del mismo. El conocimiento confidencial integral es requerido para manipular el diseño del chip de modo que la seguridad sea afectada adversamente. Por consiguiente tal ataque es improbable. Además, los institutos experimentales independientes evalúan la arquitectura y las medidas de seguridad de casi todos los semiconductores de TIs hoy en día.

4.4.1.1 Protección: Criterio de Diseño

Hay algunos criterios fundamentales concernientes al diseño de las funciones de un microcontrolador de una TI. Por un lado, las medidas en contra de los ataques estáticos y dinámicos tienen que ser efectivas. Los sensores y elementos protectores son de poco uso si pueden ser soslayados fácilmente o si

no tienen efecto bajo ciertas condiciones. Podríamos imaginar que existen sensores en un chip de semiconductor que precisan una cantidad tan grande de espacio tal que es posible destruirlos con una aguja, de este modo este mecanismo de protección quedaría deshabilitado.

Un criterio del diseño muy importante que es diferente de los criterios usualmente usados para chips estándar es el requisito de que no debe haber ningún mecanismo o función indocumentada bajo ninguna circunstancia. La mayoría de las veces tales 'características' indocumentadas no están completamente probadas toda vez que solo unos cuantos las conocen y por eso a menudo tienen un número de desperfectos o lados flacos. Como no están documentadas, no podrían ser incluidas en la evaluación del hardware y ellas posiblemente podrían servir para ataques en un momento posterior. Consecuentemente, el uso de tales características indocumentadas está estrictamente prohibido, aun si podrían ser de ayuda sustancial para los desarrolladores.

4.4.1.2 Protección: Números de CHIP ambiguos

Durante el desarrollo del semiconductor todos los elementos de garantía referidos al hardware deben ser diseñados de acuerdo al microcontrolador a ser producido. A parte de los sensores y capas protectoras, una memoria WORM (escrita una vez, leída múltiples veces) es usada. Durante la producción del semiconductor, un número inequívoco del chip se guarda en esa memoria. De esta manera el chip individualizado, puede ser claramente rastreado y la TI puede ser identificada sin ambigüedades a todo lo largo del sistema. Además, este número puede servir para obtener como resultado llaves ofreciendo la posibilidad de establecer listas que posibiliten abstraer TIs sospechosas de circulación. Debería ser notado, sin embargo, que aunque este número no puede variarse en el chip original, no hay protección de copias de este chip al haber sido hecho por un microcontrolador libremente programable. Por consiguiente, los mecanismos

de seguridad no deben basarse sobre un chip almacenando un número en su memoria WORM. Este número no ambiguo sólo puede ser la base para mecanismos de seguridad realmente criptográficos.

4.4.2 DESARROLLO DEL SISTEMA OPERATIVO DE LA TI.

El desarrollo de software para TIs se implementado paralelamente a los principios de desarrollo del software en uso en esos momentos. Ciertas condiciones tienen que ser reunidas pese a las metodologías de desarrollo (el modelo en cascada, el modelo en espiral, etc.) que son usadas.

Es esencial que las computadoras usadas para el desarrollo de tal software estén incorporadas en una red propia y completamente independiente que no permita cualquier vía de entrada desde el exterior. Las herramientas de desarrollo, como compiladores y simuladores del chip, son paquetes de software cuya funcionalidad ha sido examinada en pruebas independientes. En algunos casos, incluso dos compiladores diferentes son usados para asegurar la exactitud del resultado. Por regla general, el uso de software cuyo origen no puede ser determinado exactamente es prohibido, ya que ésta podría ser una forma posible de manipular la herramienta de desarrollo para cambiar el código de programa a ser creado.

4.4.2.1 Protección: Principios de desarrollo

Semejantes al proceso de desarrollo del hardware, las características indocumentadas no deben ser implementadas en el transcurso de desarrollo del software. Sería muy útil, por ejemplo, incluir comandos dando a cualquier área de memoria deseada permisos de ser leída para convertir las cajas negras de pruebas de consumo de tiempo típicas de TIs en cajas blancas de pruebas. Si, no obstante, una de esas órdenes fue olvidada en el programa, entonces las llaves

secretas podrían ser determinadas de las TIs reales. Para evitar tal estrategia de ataque desde el propio inicio, incluyendo cualquier comando de descarga de datos es no deseado en el programa aun cuando esto ayudaría a disminuir el costo de tiempo de desarrollo intensivo. No obstante, la presión de una fecha tope y la complejidad constantemente creciente de los Sistemas Operativos de TIs han conducido a la socavación de este principio. Para garantizar que estas órdenes que acompañan el proceso de desarrollo no están incluidas en TIs reales bajo ninguna circunstancia, se realizan pruebas especiales durante el proceso de fabricación de TIs para asegurar la inexistencia de tales órdenes.

Otro principio demanda que el desarrollo de un programa nunca debe ser efectuado por un solo programador. Esto es para cumplir con el control de calidad del software por un lado y por otro, las ediciones de seguridad referentes a los ataques también requieren un principio de doble control para ser aplicadas en cualquier momento durante el desarrollo del software.

Éste es un método muy efectivo para hacer las tarjetas más resistentes a ataques 'desde adentro' ya que al menos dos desarrolladores tienen que convenir en el proceso de desarrollo en cualquier momento. Además, las inspecciones del código fuente interno se realizan de forma regular para asegurar la calidad y para monitorear el proceso de desarrollo.

Una vez que el proceso de desarrollo de software ha sido concluido, es común que el código fuente completo sea revelado así como su funcionalidad sea examinada por institutos experimentales independientes dentro del alcance de una evaluación de software. La razón principal para estos exámenes intensivos de tiempo y costo es la exclusión de defectos de software, así también, hacen imposible que un desarrollador, por ejemplo, esconda un caballo de Troya en el Sistema Operativo.

4.4.2.2 Protección: Distribución del conocimiento

Si varias personas están trabajando en una tarea, entonces el resultado es mucho más robusto contra ataques debido a que existen opiniones y experiencias diferentes de las personas involucradas. El principio de distribución de conocimiento contrarreste la ventaja de que todo el mundo conoce todo a cerca de algo (el secreto compartido). Por principio, durante el desarrollo de componentes de seguridad, el conocimiento completo no debería estar concentrado en una persona, ya que esta persona podría llegar a ser un atacante potencial. Similar a lo que ocurre en ciertas áreas militares, el conocimiento obtenido durante el desarrollo es distribuido entre grupos diversos de personas, lo cual hace posible que expertos tengan debates relacionados con un tema mientras que nadie conoce todo al respecto. Lo mismo tiene aplicación exacta para la finalización del COS, esto es, cargando tablas, código de programa y datos de configuración en la EEPROM. Con excepción de la flexibilidad más alta, esto también constituye un aspecto de seguridad. Desde ese punto el conocimiento completo del COS yace sobre el fabricante del chip, que recibe el código de programa ensamblado completamente en la ROM para confeccionar las máscaras. Las partes del COS que están almacenadas en la EEPROM no son disponibles para el fabricante del chip, lo cual lo imposibilita de obtener conocimientos de los mecanismos completos de seguridad y de la funcionalidad del COS si analiza el código de la ROM.

4.5 ATAQUES Y CONTRAMEDIDAS DURANTE EL PROCESO DE FABRICACIÓN.

Los ataques durante el proceso de fabricación del chip de la TI son típicamente ataques desde el interior ya que el correspondiente ambiente de fabricación es un ambiente cerrado. El acceso es cercanamente regulado y cada acceso es registrado. No obstante, en el proceso de fabricación deben ser

incluidos los aspectos de seguridad porque algunos ataques técnicamente muy interesantes y efectivos pueden ser efectuados aquí.

4.5.1 PROTECCIÓN: AUTENTICACIÓN EN LOS PASOS DE FABRICACIÓN

Incluso durante la producción de hostias, los microcontroladores de TIs con individualizados por un número de chip y protegidos con un código de transporte. Con respecto a los últimos COS, cada chip tiene un código individual de transporte y la autenticación es obligatoria para cada acceso relacionado con la producción del chip. A la vez que esto hace que el proceso de producción consuma mayor tiempo y se requiere un módulo de seguridad en las máquinas correspondientes, la seguridad aumenta significativamente. Un ataque obvio durante el proceso de fabricación es la infiltración de chips o TIs falsas que se comportan idénticamente a los componentes regulares, por ejemplo, tienen, un comando para vaciar la memoria. Por supuesto, el reemplazo de un chip real con un chip falso solo es posible una vez que las hostias han sido divididas. Este tipo de ataque puede ser ilustrado por medio de una TI para firmas digitales: Durante la inicialización, el atacante remplaza una TI real con una tarjeta de prueba. Luego esta tarjeta es inicializada con datos reales y personalizada en un paso posterior. Como esta TI tiene todas las funciones de una TI real, las llaves para el algoritmo criptográfico asimétrico en el microcontrolador podrían también ser generadas. Los datos requeridos pueden obtenerse desde los datos de inicialización y personalización. A continuación, el atacante podría conseguir el acceso de su TI y podría hacer uso de su comando especial de copia de memoria para leer la llave distintiva secreta de la tarjeta. Ya que en el círculo de confianza ha firmado con la llave pública correspondiente, se ha verificado la tarjeta como válida. De esta forma, el atacante tiene toda la información necesaria para producir un número ilimitado de duplicados de una tarjeta válida.

Tal ataque es poco realista ya que las características organizacionales de los fabricantes hacen imposible que chips o TIs sean ingresados o sustraídos del

establecimiento correspondiente. Además, la autenticación entre la TI y el módulo de seguridad de la máquina manufacturera requerida para todos los pasos de fabricación impiden el reemplazo de chips o tarjetas.

4.6 ATAQUES Y CONTRAMEDIDAS DURANTE EL USO DE LA TARJETA.

En contraste a las fases del ciclo de vida previas de la tarjeta, el acceso a los componentes a ser atacados – la TI – usualmente precisa mucho menos esfuerzo de parte del atacante una vez que la TI ha sido expedida. Ésta es una de las razones por las que la probabilidad de un ataque es relativamente alta especialmente durante la fase en la cual la tarjeta es usada.

4.6.1 ATAQUES AL NIVEL FÍSICO

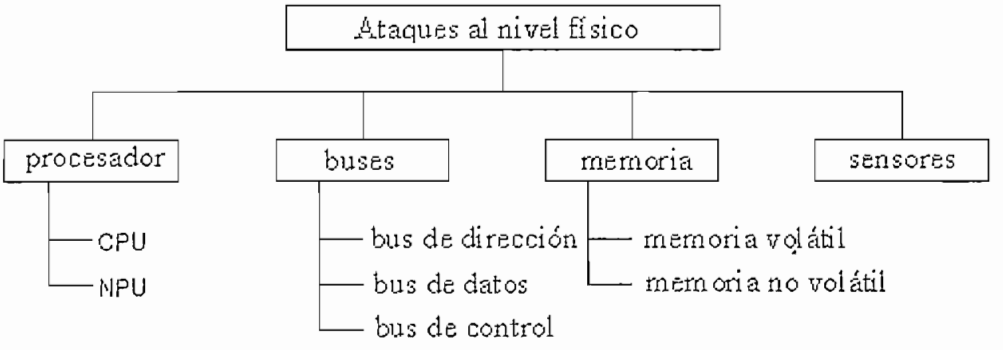

Figura 4.11 Clasificación de los puntos de ataque sobre un microcontrolador de TIs al nivel físico. (*Referencia:* [WOLFGANG])

Las manipulaciones en el área de semiconductores requieren una gran cantidad de esfuerzo técnico. Dependiendo del escenario de ataque, el equipo requerido podría incluir un microscopio, un cortador de láser, micromanipuladores y computadores muy rápidos para el análisis, registro y evaluación de los procesos eléctricos en el chip. El equipo y conocimientos para este tipo de ataques solamente lo poseen pocos especialistas y organizaciones, lo cual reduce

en mucho la probabilidad de ataques a este nivel. No obstante un fabricante de TIs o semiconductores siempre debe dar por supuesto que un asaltante potencial puede usar todo el equipo requerido para tal ataque; por consiguiente, las características de seguridad correspondientes deben ser incluidas en el hardware.

4.6.1.1 Análisis estático de los microcontroladores de TIs

4.6.1.1.1 Protección: Tecnología del semiconductor

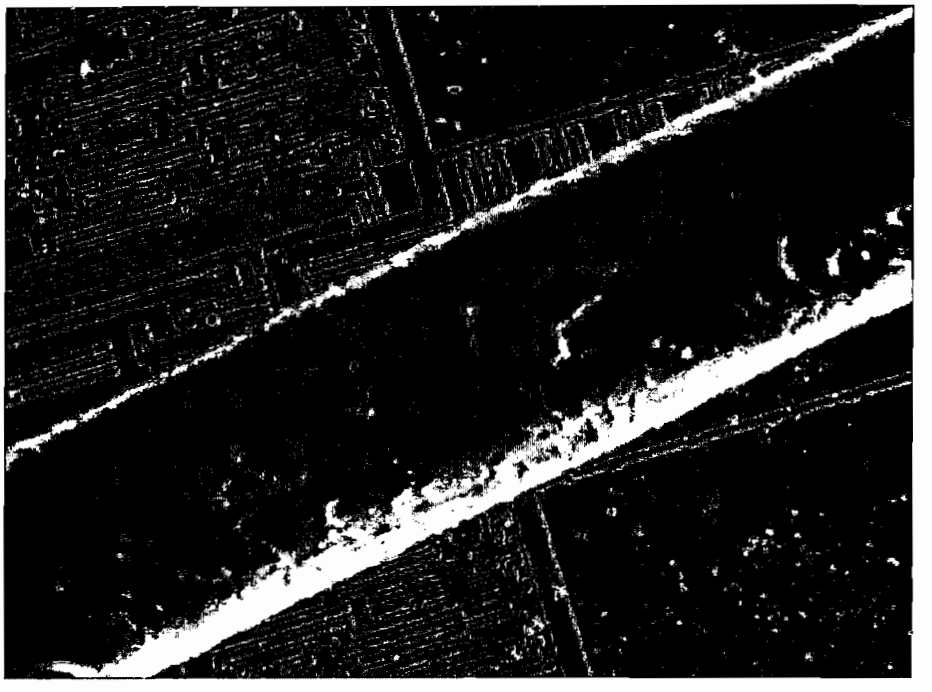

Figura 4.12 Comparación de un cabello humano con la estructura del semiconductor de un microcontrolador de TIs, magnificado 1000x. (*Referencia:* [WOLFGANG])

Las estructuras de los chips (ancho de los caminos del semiconductor, tamaño de los transistores, etc.) han alcanzado límites que son técnicamente posibles hoy en día. Las medidas de ancho típicas de las estructuras están en el rango de 0,35 µm y 0,13 µm, lo cual por si mismo no es un característica especial tecnológica. La densidad de transistor sobre el silicio, sin embargo, ha alcanzado el límite más alto que se puede alcanzar en la actualidad usando procedimientos de fabricación típicamente litográficos. Solo esas estructuras muy finas hacen

difícil obtener información del chip con la ayuda de un procedimiento analítico. Por consiguiente, la tecnología de semiconductores con tamaños estructurales de un micrómetro y menos actualmente son consideradas como seguras. En el futuro, esto ciertamente cambiará.

4.6.1.1.2 Protección: Diseño del chip

Las así llamadas celdas estándar las cuales, por ejemplo, contienen un procesador central o ciertos tipos de memoria, a menudo se usan para el diseño de componentes relacionados con semiconductores. La ventaja cae en el hecho de que los fabricantes de semiconductores pueden utilizar esos elementos estándar para producir rápidamente una variedad de diferentes chips de alta calidad. Este procedimiento fue desarrollado para la producción en masa de productos que no toman en cuenta aspectos de seguridad. Por tanto, esto no se usa en la producción de microcontroladores de TIs debido a las siguientes razones: la estructura y modo de funcionamiento de las celdas estándar es de conocimiento público, lo cual le provee a un posible atacante de mucha información que haría su trabajo más fácil.

4.6.1.1.3 Protección: Buses sobre el chip

Todos los buses internos en el chip conectan al procesador con los tres tipos diferentes de memoria, ROM, EEPROM y RAM, no se dirigen al exterior y por consiguiente no pueden ser contactados. No hay posibilidad de que un atacante intercepte o tenga influencia en los buses de direcciones, datos o control del microcontrolador para obtener conocimientos respecto del contenido de memoria. Usualmente, los buses son integrados en las capas más bajas del semiconductor, lo cual dificulta el hecho de poder contactarlos directamente desde la superficie. Adicionalmente, los buses en el chip son puestos en desorden (scrambled) ya sea estáticamente, individualmente por chip o individualmente por sesión, de modo que la función de los circuitos individuales de bus no pueden ser

determinados por fuentes externas. Existen microcontroladores de **TI**s con la característica de cambiar constantemente el proceso de scrambling de los buses incluso durante una sesión que está ejecutándose.

4.6.1.1.4 Protección: Diseño de la memoria

La memoria ROM es un medio de memoria para la mayoría de programas, El contenido de la ROM generalmente se usa para propósitos industriales y se puede leer con un microscopio luminoso bit por bit. Consecuentemente, no es difícil combinar los bits para formar bytes y luego combinar éstos para formar el código completo almacenado en la ROM. Para evitar este análisis, no se integra la ROM en las capas de arriba a las que se puede tener acceso fácilmente sino en las capas mas bajas de silicio. Esto previene que un ataque óptico pueda ser efectuado.

Si, no obstante, el lado frontal del chip es adherido a un soporte y el chip esmerilado por su lado posterior, el contenido de la ROM podría ser obtenido. Para evitar esto, los microcontroladores de **TI**s están equipados exclusivamente con ROM implementadas con la técnica ion, cuyos datos almacenados no son visibles, ni en el espectro visible, IR o UV. Esto también protege al chip para una gran extensión desde el así llamado grabado en agua fuerte selectivo. Este procedimiento se usa para grabar al agua fuerte el semiconductor de forma que el contenido de la ROM llegue a ser visible al ojo humano.

4.6.1.1.5 Protección: Capas protectoras (Escudos)

Una amenaza yace en el análisis de los potenciales eléctricos en el chip durante la operación. Si la frecuencia de escaneo es suficientemente alta, esto hace posible que los potenciales de carga sean medidos, por ejemplo, voltajes, sobre muy pequeñas áreas del cristal y de esta manera se puede sacar

conclusiones sobre los datos contenidos en le memoria RAM durante la operación. Esto se pude evitar exitosamente colocando capas conductoras metálicas sobre el área de memoria correspondiente o al chip enteramente. Si aquellas capas se remueven usando procesos químicos, el chip dejará de funcionar, ya que esas capas son necesarias para el suministro de voltaje que se requiere para el funcionamiento correcto del chip. En muchos casos, varias capas protectoras son organizadas una sobre otra y la ausencia de daño sobre ellas es chequeada constantemente.

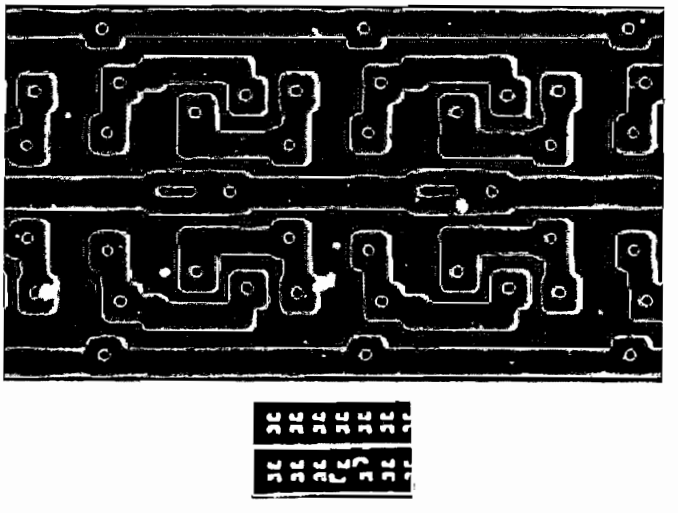

Figura 4.13 Fotografía de varias celdas RAM magnificada 3000x, sin protección de capas metálicas adicionales. La fotografía de abajo muestra los potenciales eléctricos de las mismas celdas RAM que han sido medidos usando un haz de electrones con el chip en operación. La distribución de ceros y unos en la RAM puede ser reconocida claramente. (*Referencia:* [WOLFGANG])

Además, la tecnología de semiconductor permite la implementación de estructuras de laberinto conformado transportadoras de corriente sobre la totalidad de la superficie del chip o sobre áreas que son de alto riesgo. Ellas pueden ser fácilmente monitoreadas mediante la medición de la resistencia o la capacitancia o se podría implementar funciones dentro del chip que hagan que éste se apague si se detecta que existe algún daño sobre dichas estructuras.

4.6.1.1.6 Protección: Scrambling de la Memoria

De forma similar a lo que es el scrambling de los buses, el scrambling de la memoria está siendo usado cada vez más en los chips microcontroladores. La seguridad se basa en mantener el scrambling patrón de las celdas de memoria en secreto. La memoria puede ser desordenada (scrambling) y se requiere poco espacio adicional en el chip. Sin la información de scrambling correspondiente es extremadamente difícil para un atacante determinar la forma en que las celdas de memoria son direccionadas.

incremento lineal de direcciones de memoria

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22							
30									
40									

scrambling en las direcciones de memoria

06	01	19	03	04	05	40	07	10	09
15	11	12	13	14	18	16			
20	21	22	17	00	02				
30									
08									

Figura 4.14 Comparación de una memoria convencional y una memoria desordenada usada en un microcontrolador de TI. (*Referencia:* [WOLFGANG])

4.6.1.1.7 Protección: Encripción de la Memoria

A parte del intercambio de los datos en la memoria (scrambling), los microcontroladores modernos de **TI**s ofrecen la posibilidad de encriptar la memoria ya sea una parte del registro del procesador sobre una cantidad de cosas o un nivel individual del chip. Durante este proceso, los datos correspondientes son encriptados y deencriptados en tiempo real durante la lectura y escritura. Adicional a la llave, algunos tipos de chips ofrecen la opción de incluir el direccionamiento de la memoria en el proceso de encripción/deencripción. Esto resulta en que datos iguales tienen valores diferentes en posiciones diferentes de la memoria después de la encripción. Las

llaves individuales para cada sesión pueden ser especialmente usadas en áreas de la RAM.

Si un ataque tuvo éxito y los datos fueron leídos desde la memoria, sería necesario contar con la llave pública para que la información obtenida pueda ser entendida. Esto incrementa el esfuerzo requerido por parte del atacante dado que ya sea debería conocer la ubicación de la llave almacenada o leer todos los datos disponibles en el chip.

4.6.1.2 Análisis dinámico de los microcontroladores de TIs

4.6.1.2.1 Protección: Monitoreo de la capa pasiva

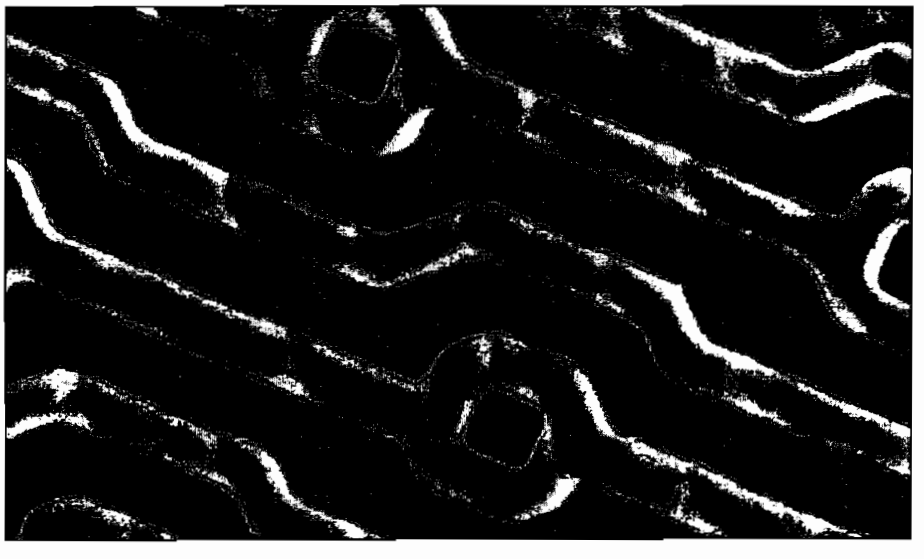

Figura 4.15 Capa pasiva moderna magnificada 8000x. La separación de pistas es 4 μm y el principio de funcionamiento de este detector se basa en la medición de resistencia. (*Referencia:* [WOLFGANG])

Después de que el microcontrolador ha sido producido sobre el silicio, se añade una capa pasiva, la cual previene la oxidación, por ejemplo por el oxígeno atmosférico, al igual que otros procesos químicos que tienen lugar en la superficie del chip. Para manipular el chip, esta capa pasiva siempre debe ser removida en primer lugar. Sin embargo, se debería tomar en cuenta que la capa pasiva puede

ser removida químicamente, por lo que el chip está sujeto a un alto riesgo de oxidación, lo cual puede destruirlo rápidamente. Un circuito sensor puede determinar mediante la medición de resistencia o capacitancia si la capa pasiva se encuentra en cada momento en su lugar. Si la capa no está o si sufrió algún daño, una interrupción en el software del chip puede ser disparada o el chip por completo podría ser desconectado del hardware, lo cual previene confiablemente todo análisis dinámico.

4.6.1.2.1 Protección: Control de voltaje

Cada microcontrolador de TIs está equipado con un sistema de control de voltaje. Este sistema es responsable del apagado de los componentes en un microcontrolador de una manera controlada cuando se exceden los límites superior o inferior del voltaje de operación. Esto asegura que el software nunca operara en los límites de voltaje en los cuales el chip podría sufrir algún daño. Si no se contase con el control de voltaje, una operación en esos rangos límite podría, por ejemplo, conducir a que el contador del programa no se ejecute de manera estable por mas tiempo, lo cual conduciría por ejemplo a que se den saltos no controlados en el programa o podría causar cálculos equivocados en el procesador. Este comportamiento anormal podría ser usado como punto de partida para la determinación de llaves secretas por el método del análisis diferencial de fallas (DFA).

4.6.1.2.2 Protección: Monitoreo de frecuencia

Generalmente la tasa central da la TI se regula externamente, así la velocidad del reloj interno se determina desde afuera. Esto ofrece la posibilidad – al menos teóricamente – de ejecución del microcontrolador en operación de un solo paso. Esto podría conducir a la existencia de excelentes posibilidades de realizar un análisis especialmente respecto de la medición del consumo de energía y la determinación de potenciales eléctricos en el chip. Para evitar este

tipo de ataque, un funcional grupo de componentes para la detección de sobre o sub frecuencias está integrado en el chip. Esto impide que la tasa fundamental definida sea disminuida a un valor inadmisible.

Para proteger al microcontrolador del peligro de ser manejado en el modo de paso único es significativo asegurar los detectores de baja frecuencia con capas protectoras, a fin de que el microcontrolador no pueda ser manipulado de forma inadvertida.

4.6.1.2.3 Protección: Scrambling de los buses

Muchos microcontroladores de TIs realizan un scrambling de los buses de direccionamiento de memoria, los cuales son solamente accesibles internamente en el chip. Esto significa que los circuitos de bus individuales no están organizados en un orden ascendente o descendente sino en un orden confuso y muchas veces intercambiado con respecto de los demás o incluso aislado por medio de capas que separan uno de otro. Éste es un obstáculo para atacantes potenciales, ya que ellos no conocen cuál circuito de buses tiene cuál función o dirección.

Originalmente este scrambling de los circuitos de buses fue introducido solamente en una variante estática. Es decir, con scrambling idéntico en cada chip. De este modo, a mediano plazo no sería un problema real para un atacante descubrir cómo se realiza el scrambling en el circuito y consecuentemente podría iniciar una campaña de bugging.

Sin embargo, existe una mejora en la seguridad al introducir procesos de scrambling de los buses para cada chip individual. Este scrambling individual para cada chip no es realizado por la producción de diferentes máscaras de exposición par los buses de cada chip ya que no puede ser realizado técnicamente al

momento y podría resultar muy costoso. El scrambling se realiza por scramblers, los cuales están localizados directamente en la memoria y pueden ser controlados por los números de chip individuales. Preferentemente este procedimiento puede ser efectuado sin ningún esfuerzo con semiconductores y esto hace que una campaña de bugging sea mucho más difícil. Un proceso de scrambling que es individual para cada chip y cada sesión puede ser realizado adecuadamente usando valores variables de entrada en los scramblers.

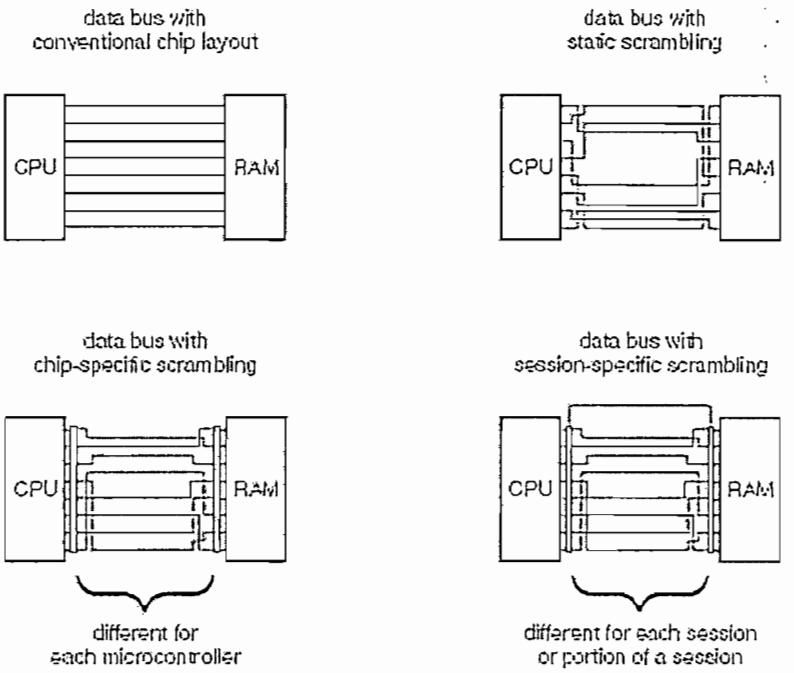

Figura 4.16 Scrambling de buses en un microcontrolador de TIs, ilustrando el uso de un bus de datos de 8 bits entre el CPU y la RAM. Las líneas del bus de datos mostradas aquí representan los flujos de información. (*Referencia:* [WOLFGANG])

4.6.1.2.4 Análisis dinámico y defensa: Medición del consumo de energía del CPU.

En junio de 1998 Paul Kocher, Joshua Jaffe y Benjamín Jun publicó un documento sobre un análisis de energía simple (SPA) y análisis diferencial de energía (DPA).

El principio del análisis de energía simple (SPA) es muy simple. Un convertidor análogo-digital se usa para medir el consumo de energía de un

microcontrolador por la determinación de la caída de voltaje en un resistor conectado en serie a una resolución temporal alta. Dada la estructura relativamente simple de los CPUs de microcontroladores de TIs los procesos internos y los datos procesados conducen a que se pueda medir e interpretar el consumo de poder. Para hacerlo más claro, uno puede imaginar que la misma secuencia con los mismos datos conducen a un cierto ciclo de consumo de energía del procesador. Si este programa corre con otros datos el ciclo de consumo de energía difiere. Esta desviación se usa para determinar los datos procesados.

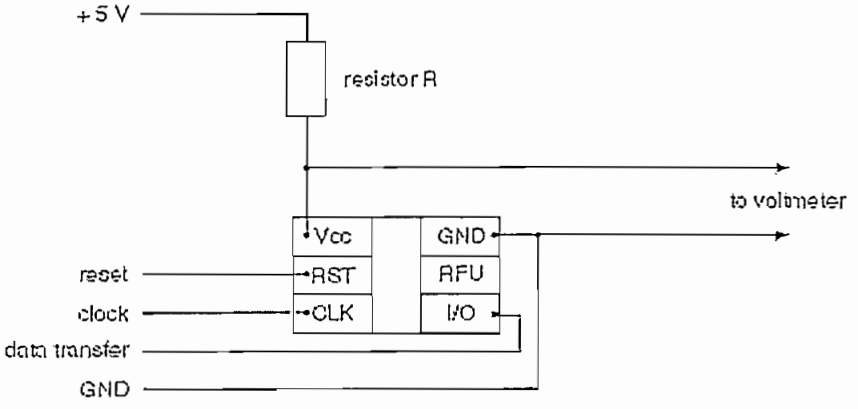

Figura 4.17 Diagrama del circuito de conexiones a un microcontrolador de TIs para hacer mediciones simples de corriente usando resistencias en serie. (*Referencia:* [WOLFGANG])

En comparación al análisis simple de energía (SPA) el análisis de energía diferencial (DPA) hace posible descubrir incluso las diferencias más pequeñas en el consumo de energía del microcontrolador. Para este propósito, el consumo de energía se lo determina primero durante el procesamiento de datos conocidos y luego durante el procesamiento de datos desconocidos. La medición es generalmente repetida varias veces y el valor más probable es calculado para eliminar el ruido. Después que la medición está terminada, se determina la diferencia y del resultado el dato desconocido puede ser inferido.

El análisis de energía de los microcontroladores de TIs es un ataque que debe ser tomado muy en cuenta para hardware y software que no está preparado. La razón para esto es que en algunos microcontroladores podrían existir dependencias del consumo de poder desde la correspondiente instrucción de

máquina y también de los datos procesados con estas instrucciones de máquina. Además, el esfuerzo requerido para un ataque exitoso referente a equipo de medición es muy pequeño. Sin embargo, existe un número de contramedidas efectivas, las cuales se basan en mejorar el hardware por un lado y modificar el software por otro.

La solución más simple relacionada con el hardware es la instalación de un rápido regulador de voltaje sobre el chip que asegure que el consumo de energía sea independiente de las instrucciones de máquina y los datos con la ayuda de un resistor de desviación. La inserción de una fuente de ruido generado al azar sobre el chip también es una solución efectiva. Una solución técnicamente más retadora es un semiconductor de diseño modificado del procesador que conduzca a un consumo constante de energía. Sin embargo, algunos de estos intentos incrementan el consumo de energía de los microcontroladores lo cual no es deseable en ciertas áreas de aplicación como el sector de telecomunicaciones. Una contramedida simple durante procesos críticos SPA/DPA también puede ser la activación de componentes no requeridos para este propósito tales como generadores de sumas de verificación CRC o coprocesadores numéricos con datos generados al azar como entrada para producir ruido artificial de consumo de energía.

El uso de estados de espera generados al azar en el procesador complica significativamente los procesos de sincronización durante el análisis de consumo de energía sin tener la desventaja de un consumo de energía más alto.

Con respecto a las contramedidas referentes al software existe un amplio rango de soluciones hasta el momento. El acercamiento más simple a una solución es el uso exclusivo de instrucciones de máquina con consumos de energía muy similares. Las instrucciones de máquina con desviaciones significantes del consumo de energía promedio ya no deben ser usadas en código assembler. Otra medida es la introducción de órdenes diferentes para los mismos

cálculos de cripto-algoritmos que se escogen al azar. Para el observador es mucho más difícil reconocer la convergencia entre instrucciones de máquina y datos procesados conocidos y desconocidos. Una medida similar es utilizar tablas individuales por cada chip para las S-boxes del algoritmo (triple) DES.

Para complicar la adquisición de datos que requiere un ataque de análisis de energía exitoso, todas las llaves deberían estar aseguradas por la irreversibilidad de los contadores de reintentos. Además, es necesario bloquear el acceso libre a todos los comandos del tipo INTERNAL AUTHENTICATE en los cuales algún dato puede ser enviado a través de un cripto-algoritmo de la TI.

4.6.1.2.5 Análisis y defensa: Medición de la radiación electromagnética del CPU.

Al menos teóricamente se pueden sacar conclusiones de la radiación electromagnética sobre las secuencias internas de eventos del microcontrolador de la TI, lo cual es un proceso similar al análisis de energía diferencial. Los SQUIDs (*superconducting quantum interference devices*) se pueden usar para medir el campo magnético de baja extensión y fuerza. La evaluación se la puede realizar paralelamente a SPA/DPA. Sin embargo, el esfuerzo técnico requerido es alto y generalmente no se dispone de los conocimientos de las estructuras internas del semiconductor. Adicional a esto, los componentes del semiconductor pueden ser protegidos muy efectivamente de este tipo de ataques por la disposición de varios caminos conductores en la superficie superior de cada uno de ellos así el campo magnético se puede determinar con detectores sensitivos pero no la conducción que lleva la corriente.

4.6.2 ATAQUES AL NIVEL LÓGICO

Los ataques contra la seguridad de una TI a un nivel lógico sobre todo requieren de un entendimiento de las comunicaciones y el flujo de información

entre el terminal y la TI. No es muy importante entender los procesos a nivel de hardware sino entender los procesos de software. Desde un punto de vista de tecnología de la información los escenarios de ejemplo descritos aquí son considerados para estar un nivel más arriba que los ataques que hacen uso de las propiedades de hardware.

4.6.2.1 Ataque y defensa: Tarjeta inteligente falsa

El ataque más concebible es el uso de una TI que ha sido auto programada y enriquecida con algunas funciones de análisis y protocolo. Hoy, las TIs y los programas de configuración pueden ser comprados libremente de cualquier compañía. Esto incrementa el curso del número de posibilidades disponibles para un atacante. Aun sin esto, con cierta cantidad de esfuerzo y destreza es posible ensamblar una TI funcional usando una tarjeta de plástico y un microcontrolador estándar en un paquete SMD. Este tipo de tarjeta puede al menos estar hecha para imitar la interfaz eléctrica de una TI real y comportarse de la misma forma para la transferencia de datos. Ahora es posible obtener tales tarjetas desde una amplia variedad de fuentes vía Internet.

Con este tipo de TIs falsas, sería posible al menos grabar un parte de las comunicaciones con un terminal y subsecuentemente evaluar esta información. Después de varios intentos, probablemente sería posible realizar parte de las comunicaciones de forma exacta a la de una TI genuina. El hecho de que esta tarjeta pueda tomar alguna ventaja es dudoso, ya que todas las aplicaciones profesionalmente diseñadas tienen protección criptográfica para las actividades importantes. Mientras no se conozca la llave secreta, el ataque no podrá ir más allá de la primera autenticación. Este tipo de ataques solo pueden ser exitosos si se conoce la llave secreta o la aplicación corre completamente sin ningún tipo de protección criptográfica. Es altamente dudoso que algún beneficio que pudiera obtenerse de este tipo de ataque sea lo suficientemente alto como para satisfacer el esfuerzo que demanda.

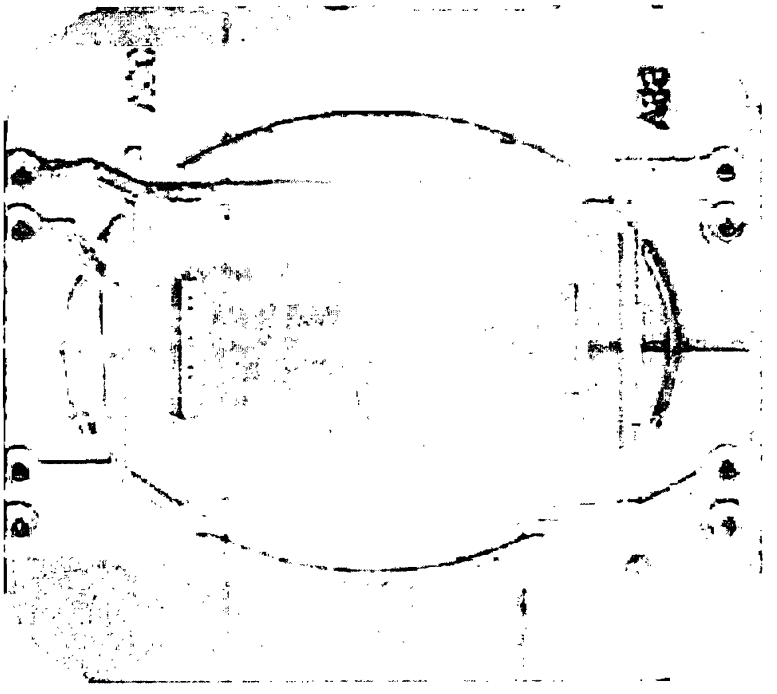

Figura 4.18 Vista de la parte posterior de un módulo de TI abierto, el chip de la izquierda es un microcontrolador PIC que está conectado a una memoria EEPROM a la derecha por cables y caminos. Este tipo de módulo es usado típicamente para clonar una TI y otros tipos de ataques sobre sistemas de TIs. (*Referencia:* [WOLFGANG])

4.6.2.2 Ataque y defensa: Interrupción de energía

	purse balance	purse balance file (in binary notation)
1. current purse balance	100 EUR	°0110 0100°
2. deduct 10 EUR	90 EUR	
3. erase the EEPROM	255 EUR	°1111 1111°
4. write the new purse balance	90 EUR	°0101 1010°

Figura 4.19 Ejemplo del procedimiento de un nuevo balance en un monedero electrónico. (*Referencia:* [WOLFGANG])

Referente a la figura 4.19, se asume que el estado de borrado de una EEPROM es un 1 lógico. Debido a la forma en que trabaja la EEPROM, esto significa que la página de la EEPROM debe ser borrada completamente (lo que significa poner todos los bits en 1) si se desea cambiar solo un bit de 0 a 1. En este ejemplo, si

se corta la energía de la TI exactamente después de que la EEPROM ha sido borrada, esto es después del paso 3, el valor del balance sería cargado al máximo y el atacante habrá creado efectivamente dinero.

Un tipo de ataque que fue realizado en TIs de manera exitosa hasta no hace mucho es interrumpir la energía a la tarjeta en un momento particular mientras un comando está siendo ejecutado. Este tipo de ataque se basa en el hecho de que con programación convencional, todas las operaciones de escritura en las páginas de la EEPROM son realizadas secuencialmente. Si el programador no ha sido listo al disponer el orden de las operaciones de escritura, un atacante puede obtener una ventaja apagando la energía al mismo tiempo.

Los diseñadores de COS conocen una contramedida efectiva para este tipo de ataque, la cual es el uso de operaciones atómicas. La característica de una operación atómica es que es indivisible, lo que significa que esta operación es realizada completamente o no del todo.

4.6.2.3 Ataque y defensa: Análisis de energía por comparación del PIN

Un tipo de ataque técnicamente muy interesante sobre la comparación de características, tales como PINs, puede ser ejecutado usando una combinación de mediciones físicas de un parámetro y variación de valores lógicos. Este tipo de ataque relaciona a todos los mecanismos en los cuales los datos son enviados a la TI y los compara en la tarjeta con los valores correspondientes, con un contador de reintentos que se va incrementando de acuerdo al resultado de la comparación.

Existen dos métodos básicos para evitar este ataque. La defensa más simple consiste en incrementar siempre el contador de reintentos antes de hacer la comparación y luego decrementarlo según sea el caso. En este caso, el

atacante no puede obtener una ventaja, independientemente de cuando él interrumpe la energía a la tarjeta, dado que el contador de reintentos ya ha sido incrementado. El segundo método de defensa es más complicado, pero provee similar protección. En esta aproximación, el contador de reintentos es incrementado después de una comparación negativa y escrito en una celda que no está usada en la EEPROM después de una comparación positiva. Los dos accesos de escritura ocurren al mismo tiempo en el proceso, por tanto el atacante no puede sacar conclusiones con respecto al resultado de la comparación. El aprende el resultado de la comparación solo después de recibir el código de retorno y en este punto es muy tarde evitar un acceso de escritura al contador de reintentos por corte de energía.

4.6.2.4 Ataque y defensa: Análisis de tiempo por comparación del PIN

Los programadores siempre dan atención considerable para que sus programas se ejecuten tan rápidamente como sea posible. Normalmente, ésta también es una consideración importante. Sin embargo, el hecho de que el tiempo de ejecución de un proceso haya sido optimizado puede ser utilizado para un ataque que definitivamente tiene una buena oportunidad de éxito. Si un PIN se envía a la TI para comparación, la rutina de comparación asociada normalmente compara el PIN recibido con el valor del PIN almacenado byte por byte.

Un programador quien no está conciente de seguridad programará esta rutina talque la primera diferencia entre los dos valores comparados cause que la rutina termine inmediatamente y retorne la llamada del programa. Esto conduce a variaciones de minutos en el tiempo de ejecución del proceso de comparación, lo cual no obstante puede ser medido usando el equipo adecuado. El atacante puede usar esta información para determinar el código secreto PIN de una manera relativamente rápida y efectiva.

Hasta hace pocos años atrás, éste aun era un efectivo tipo de ataques sobre las TIs. Sin embargo, hoy es un tipo de ataque conocido y las rutinas de comparación son construidas tal que todos los dígitos de un PIN siempre son comparados. Consecuentemente, no existe tiempo de diferencia entre un resultado positivo o negativo luego de la comparación.

4.6.2.5 Ataque y defensa: Algoritmos criptográficos libres de ruido

La seguridad en aplicaciones de TIs se basa en el uso de llaves secretas con algoritmos criptográficos. Naturalmente la autenticación del terminal por la tarjeta representa un objetivo atractivo para un atacante.

La tarjeta inteligente identifica al terminal enviándole un número generado al azar, el cual el terminal luego encripta y devuelve a la tarjeta. La TI luego realiza la misma encripción y compara el resultado con el valor recibido del terminal. Si los dos valores coinciden, el terminal ha sido autenticado y éste recibe un correspondiente código de retorno. Si la autenticación falla, la tarjeta envía un diferente código de retorno. El punto de partida para el atacante es analizar el tiempo de procesamiento entre el momento en el cual el comando es enviado y el momento en el cual la respuesta es devuelta por la TI.

En principio, un análisis de tiempo es una amenaza muy peligrosa para la seguridad de una TI. Sin embargo, dado que este tipo de ataque ha sido conocido por relativamente mucho tiempo, todas las tarjetas inteligentes de hoy en día usan solamente algoritmos criptográficos libres de ruido, los cuales son algoritmos para los cuales el tiempo requerido para la encripción y deencripción es independiente de los valores de entrada. Esto bloquea este tipo de ataque. Sin embargo, el programador tiene conflicto de intereses al respecto, dado que un algoritmo libre de ruido usualmente requiere más código de programa y siempre es más lento que una versión ruidosa. La razón para esto es que un algoritmo libre de ruido debe estar diseñado de forma que el camino a través del programa tenga la

misma longitud para todas las combinaciones de datos en texto plano, datos en texto cifrado y llaves.

Esto significa que el camino más largo necesariamente es el valor de referencia y todos los demás caminos deben ser modificados adecuadamente para coincidir con esta longitud.

4.6.2.6 Manipulación: Análisis diferencial por defecto (DFA)

Como es bien conocido, la operación de dispositivos electrónicos puede ser afectada adversamente al exponerlos a interferencias electromagnéticas. Por ejemplo, un teléfono móvil puede causar que los procesadores de muchos tipos de aparatos controlados por pequeñas computadoras colapsen. La causa cae en las celdas de memoria, cuyo contenido puede ser alterado por los campos AC de alta frecuencia.

El principio básico de este tipo de ataques es relativamente simple. En el primer paso, un texto plano arbitrario es encriptado usando la llave que va a ser rota y el texto cifrado resultante es almacenado. A continuación de esto, la operación de la tarjeta es alterada mientras está procesando el algoritmo criptográfico, por ejemplo, por exponerlo a radiaciones de ionización o campos de alta frecuencia para alterar un solo bit de la llave en una locación al azar mientras los cálculos están siendo realizados. Esto da como resultado un texto cifrado que está encriptado incorrectamente, debido al bit alterado. Este proceso se repite muchas veces y todos los resultados son almacenados para el análisis. El resto del procedimiento para determinar el valor de la llave secreta es puramente matemático.

La defensa más simple es simplemente calcular el algoritmo de criptografía dos veces y comparar los dos resultados. Si ellos coinciden, ningún intento ha

sido hecho para alterar algún bit desde fuera de la tarjeta. Este método de defensa asume que los errores al azar intencionalmente introducidos nunca pueden alterar el mismo bit dos veces en una fila. Esto se aplica primeramente para ataques sobre procedimientos criptográficos asimétricos de tiempo intensivo, tales como RSA y DSS.

Otra medida defensiva efectiva contra DFA puede ser efectuada al encriptar siempre diferentes textos planos. La solución más simple es para fijar el texto plano a ser encriptado con un número generado al azar. Esto significa que el algoritmo criptográfico siempre encripta datos diferentes, lo cual evita que se use DFA.

4.6.2.7 Ataque y defensa: Perturbando el procesador

Un tipo de ataque que es similar al uso de DFA para atacar la llave secreta de un algoritmo criptográfico consiste en intentar afectar la ejecución del código de programa perturbando la operación del procesador.

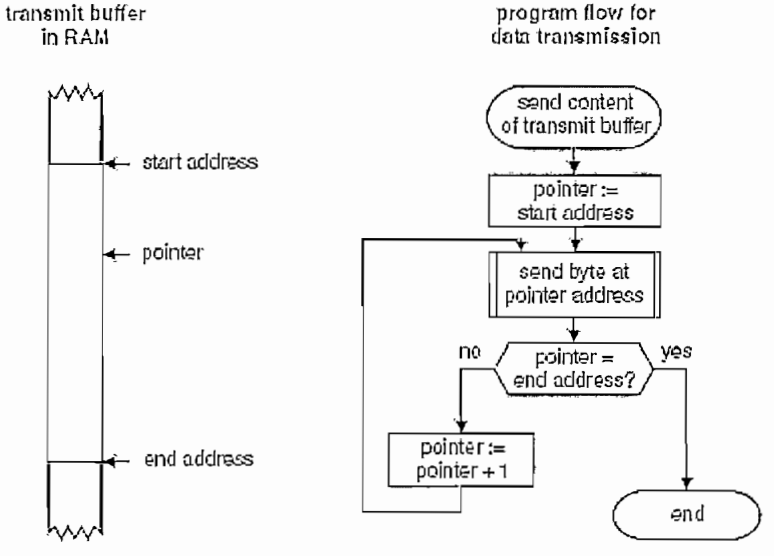

Figura 4.20 Ejemplo de una rutina no robusta para el envío del contenido de un buffer de transmisión, la cual puede ser atacada exitosamente por la perturbación del procesador. (*Referencia:* [WOLFGANG])

La operación del procesador puede ser perturbada por la aplicación de interferencias a las líneas de suministro, exponiendo el chip a destellos de luz o usando radiación de alta frecuencia, entre otras cosas. Si la perturbación es disparada en el instante apropiado durante la ejecución del programa, puede ser utilizada para influenciar intencionalmente una operación de búsqueda. Un simple ejemplo de esto se muestra en la Figura 4.20. La tarea de la rutina ilustrada es enviar el contenido de un buffer de transmisión, cuyos límites están especificados por una dirección de inicio y una dirección de fin. Si el atacante tiene éxito en perturbar intencionalmente la búsqueda que determina la dirección final del buffer de transmisión, los datos que están después del final del buffer de transmisión también son enviados al terminal. El espacio de trabajo para un algoritmo de criptografía podría estar localizado en esta región, de esta forma sus llaves podrían ser leídas ilícitamente.

La defensa contra estos ataques involucra varios niveles del sistema. Al nivel de hardware, es importante para el microcontrolador de la TI tener sensores adecuados, a fin de que pueda detectar todos los intentos de desestabilizar el procesador. Estos sensores pueden incluir detectores de interferencias de voltaje y un número grande de sensores de luz. Para hacer imposible que unos pocos sensores de luz sean cubiertos con tinta negra para neutralizarlos, una buena idea es usar un número relativamente grande de sensores distribuidos sobre la superficie del chip. Esto por sí mismo es suficiente para excluir muchos tipos de ataque. Un material opaco de encapsulación del chip provee sólo protección limitada, ya que puede ser removido de forma relativamente fácil usando métodos químicos.

Capítulo 5

Aplicaciones de las tarjetas inteligentes

5.1 TARJETAS INTELIGENTES EN LOS SISTEMAS DE PAGO

La aplicación original de las tarjetas con microcontroladores fue la de servir para la identificación de usuarios en el sector de las telecomunicaciones. Recientemente, sin embargo, las TIs se han establecido en otros sectores de mercado, conocidos como sistemas de pago electrónico. Dado al gran número de tarjetas en uso, el mercado potencial de este sector es enorme. Las aplicaciones futuras del dinero electrónico incluyen reemplazar el significado convencional del pago (billetes bancarios y monedas), compras mediante redes globales y tv del tipo pague por ver.

Las TIs son adecuadas por su naturaleza para las aplicaciones de sistemas de pago, dado que ellas pueden almacenar datos de manera fácil y segura y su tamaño conveniente y robustez hacen que sea fácil de usar para todos. Los sistemas de pago electrónico y el dinero electrónico ofrecen beneficios significativos a todos los involucrados. Para los bancos y comerciantes, reducen el costo asociado con el manejo de efectivo. Los monederos electrónicos que están fuera de línea eliminan en gran medida los costos de telecomunicaciones de datos para las transacciones de pago. Se reduce el riesgo de hurto y vandalismo, dado que los sistemas electrónicos no contienen dinero que pueda ser robado. Para los comerciantes, el hecho de que todas las transacciones sean realizadas rápidamente también es un argumento persuasivo, dado que esto significa que la administración de dinero puede ser optimizada. El dinero electrónico puede ser transferido mediante cualquier canal de telecomunicaciones deseadas, así no es necesario recolectar regularmente el dinero desde las máquinas.

Los clientes también se benefician de los nuevos métodos de pago, aunque en menor grado. No siempre será necesario tener dinero en la mano y es posible realizar pagos rápidamente a una máquina de ventas o un dispensador de boletos.

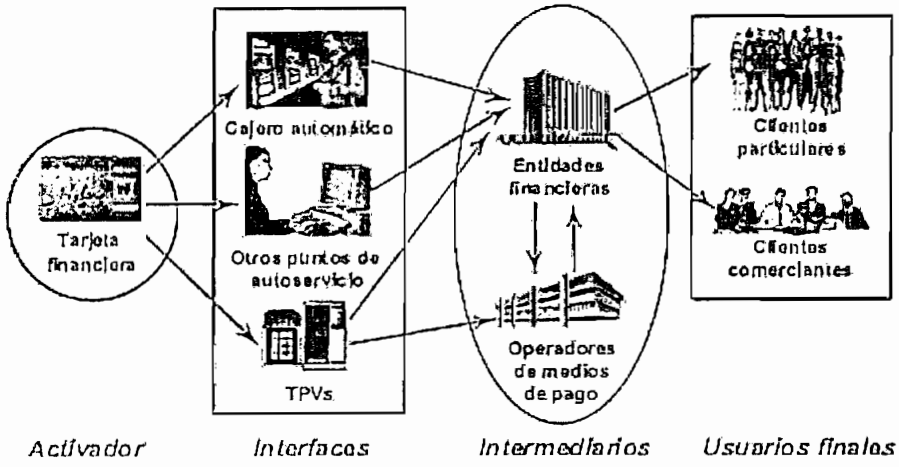

Figura 5.1 Esquema sencillo de un medio de pagos actual. (*Referencia:* [LIBURUTEGIA])

5.1.1 TRANSACCIONES DE PAGO USANDO TARJETAS

La aproximación más simple al uso de tarjetas en sistemas de pago es el uso de tarjetas con banda magnética que mantienen datos para la autorización en línea. Luego de que la tarjeta del usuario ha sido chequeada contra las listas negras y la solvencia ha sido verificada, los fondos pueden ser transferidos directamente desde la cuenta bancaria del portador de la tarjeta a la del comerciante. Con TIs el escenario es algo diferente, pero en principio mantiene lo mismo. Lógicamente se prefiere una TI para una cuenta bancaria y después de una autenticación unilateral o mutua en segundo plano del sistema y la tarjeta, una cantidad previamente ingresada es transferida. Naturalmente, la verificación de PIN también se realiza en la TI o en el sistema en segundo plano durante la transacción.

Existen tres modelos fundamentales para el pago electrónico usando TIs: (a) tarjetas de crédito, en las cuales el pago se hace después de que un servicio es suministrado (pague después), (b) tarjetas de débito, en las cuales el pago se hace cuando el servicio es suministrado (pague ahora) y (c) monederos electrónicos, en los cuales el pago se hace antes de que el servicio se suministre (pague antes).

Tarjetas de crédito

La idea original de usar una tarjeta de plástico para el pago de bienes o servicios llega con la tarjeta de crédito. El principio es simple: UD paga usando la tarjeta y la cantidad correspondiente mas tarde es debitada de su cuenta. El costo de este proceso es cargado por el comerciante, quien usualmente paga una retribución que depende de la cantidad de transacciones. Esta retribución usualmente es alrededor de 2 a 5 % del precio de compra.

Hasta ahora, la mayoría de las tarjetas de crédito no tienen chips incluidos. La desventaja de este tipo de tarjetas es que tienen un nivel de protección relativamente bajo contra falsificaciones. Consecuentemente, los emisores de tarjetas experimentan perdidas significativas debido a tarjetas falsificadas, ya que el comerciante garantizó el pago. Evidentemente, las tarjetas de crédito serán probablemente dotadas de chips en un futuro cercano, para reducir el costo firmemente creciente del fraude.

Tarjetas de débito

El país donde más se usan tarjetas de débito es Alemania. Una tarjeta de débito, la cual puede ser ya sea una tarjeta de banda magnética o un TI, permite que la cantidad de dinero que va a ser transferida a la cuenta del comerciante o proveedor del servicio como una parte directa del proceso de pago, Con ambas,

tarjeta de crédito y tarjeta de débito, el proceso actual de pago normalmente es autorizado por una verificación de crédito mediante un sistema en segundo plano. Usualmente existe un nivel de umbral sobre el cual esto debe ocurrir, de esta forma no siempre es necesario hacer una conexión con el sistema en segundo plano para compras pequeñas. El nivel de umbral está en el orden de los $ 200.[1]

Monederos electrónicos

Con un monedero electrónico, el 'dinero electrónico' se carga en la tarjeta antes de que cualquier pago sea hecho. Esto se puede hacer para un intercambio por efectivo o usando un proceso libre de efectivo. En la actualidad cuando se realiza una compra el balance en la tarjeta se reduce de acuerdo al costo del artículo comprado y en el mismo momento el balance del monedero electrónico de la segunda parte (que usualmente es el comerciante) se incrementa con la cantidad correspondiente. El comerciante mas tarde puede ceder el dinero electrónico recibido en esta manera al operador del sistema del monedero electrónico y ser acreditado con la cantidad correspondiente de dinero real. El usuario de un monedero electrónico de este modo intercambia dinero real por una forma electrónica de dinero que se carga en su TI. Cuando se hace una compra, el portador de la tarjeta intercambia este dinero electrónico por bienes o servicios.

Este sistema tiene tres desventajas significativas para el usuario. La primera es que cuando la tarjeta es cargada, el usuario recibe dinero electrónico en intercambio por el dinero real. Financieramente, de este modo el usuario da al operador del sistema monedero un préstamo sin intereses, ya que podría tomarle varias semanas al usuario gastar el dinero electrónico, mientras que el dinero real llega a ser de propiedad del operador del monedero inmediatamente. La cantidad de interés podría ser pequeña para un usuario individual, pero en total representa una sustancial fuente suplementaria de ingresos para el operador del sistema monedero.

[1] Tomado del articulo de aplicaciones pdf

Una segunda desventaja es que un problema real surge si el operador del monedero cae en bancarrota. Esto es porque el usuario de la tarjeta ha intercambiado dinero real, cuyo valor está garantizado por el estado dentro de ciertos límites, por dinero electrónico en una TI. Si el operador del monedero cae en banca rota, el dinero electrónico puede repentinamente volverse sin valor y el usuario habrá perdido su dinero.

Aun existe una significativa desventaja para el usuario. ¿Qué puede el poseedor de un monedero electrónico hacer si éste no trabaja por más tiempo? Si el monedero es anónimo, ni aun el operador del sistema monedero puede determinar la cantidad de dinero que fue cargada previamente en la tarjeta. El dueño del monedero podría encontrarse con que es prácticamente imposible proveer pruebas convincentes de cuánto dinero está todavía en la tarjeta. Si el chip es maltrecho, de esta forma el dinero electrónico se pierde irrevocablemente. Desafortunadamente, una TI es mucho menos robusta que los billetes bancarios o monedas, por razones entendibles.

5.2 TARJETAS INTELIGENTES EN TELECOMUNICACIONES

Las telecomunicaciones o tecnología de las comunicaciones, es la tecnología usada para el intercambio de mensajes a distancias arbitrarias entre personas o máquinas. Recientemente las telecomunicaciones han llegado a ser verdaderas aplicaciones masivas como el resultado de varias condiciones favorables, tales como la desregularización del mercado, la fabricación no tan cara de los equipos terminales y un boom general económico y técnico.

Uno de los primeros sistemas de primera generación en telefonía móvil en soportar el uso de tarjetas fue C-Netz (Alemán), primero fueron usadas tarjetas de banda magnética las cuales rápidamente fueron suplantadas por TIs. Esto condujo a una distinción entre el teléfono y el subscriptor, con el resultado de que

la personalización de teléfonos, la cual fue estándar en ese tiempo, no era ya necesaria. Como una consecuencia, los teléfonos, los cuales son muy caros comparados con las TIs, llegaron a ser fácilmente intercambiables. Dada la tremenda competencia, los precios de los teléfonos móviles bajaron rápidamente, lo cual fue en rédito de los operadores de red con miramiento a tener un mercado potencial más grande. El resultado final de varios años de trabajo fue una especificación para el Sistema Global para Comunicaciones Móviles (GSM). A inicios de 1985 la Unión Internacional de Telecomunicaciones (ITU), con ojo al futuro, se inicio el trabajo sobre la unificación internacional y extensión funcional de todos los sistemas de telecomunicaciones móviles. Sin embargo, éste falló al producir una solución internacional estándar, como originalmente se deseó, así en 1995 esto fue convertido en el concepto IMT-2000 ('International Mobile Telecomunication a 2000 MHz'). IMT-2000 provee considerablemente mas espacio para realizar maniobras con respecto a la implementación y vino a formar la base del actual UMTS y otros sistemas de tercera generación de telefonía móvil.

Un sistema de segunda generación (2G) se entiende que es una red de telecomunicaciones móviles celular con transmisión de datos digitales sobre el interfaz aéreo. Los dos sistemas 2G más usados son GSM y CDMA. Las extensiones funcionales de GSM, tales como el GRPS y EDGE, los cuales se dirigen en dirección de la tercera generación, son típicamente calificados como sistemas 2.5G. La tercera generación (3G) también abarca las redes de telecomunicaciones móviles celulares con interfaces de aire digitales, pero con extensiones mejores con respecto a las comunicaciones móviles de datos y servicios compatibles con Internet comparado con los sistemas 2G. Algunos sistemas 3G típicos son UMTS y CDMA 2000. Esto dos sistemas son miembros de la familia IMT-2000.

La tarjeta inteligente usada en los teléfonos móviles GSM, el cual es llamado SIM 'módulo de identidad del suscriptor', fue y aun es el pionero en términos de funcionalidad y capacidad de memoria. Esto es en parte debido al

hecho de que las TIs usadas en teléfonos móviles, cuyos costos de fabricación son de varios cientos de dólares, son significativamente menos sensitivos del costo que otros tipos de TIs, tales como aquellas usadas para el pago electrónico o aplicaciones médicas. Otro factor decisivo con respecto a la tecnología de las TIs es generalmente su alta tasa de evolución del sector entero de las telecomunicaciones.

El SIM es un módulo de seguridad obligatorio localizado en el teléfono móvil de un sistema GSM como un componente intercambiable. Esto está definido en la especificación GSM 02.17: 'la SIM es una entidad que contiene la identidad del subscriptor. La función primaria del SIM es para asegurar la autenticidad de la estación móvil con respecto a la red'.[2]

Además sus funciones primarias de soportar la identidad del subscriptor, lo cual es realizado usando un PIN y la autenticación de la estación móvil con respecto a la red, el SIM también realiza un número de otras funciones. Esto permite programar ejecución para ser protegido contra la manipulación y hace posible almacenar datos como números de discado, mensajes cortos y los ajustes personales de configuración para el teléfono móvil. Además, es el portador de servicios suplementarios de seguridad usados con telecomunicaciones móviles.

Dos formatos de SIM son usados en el sistema GSM. En teléfonos móviles diseñados para permitir que el SIM sea intercambiado relativamente a menudo, se usa el formato ID-1. Esto se basa en la idea de una compañía o familia de teléfonos con una tarjeta separada para cada usuario. Los teléfonos móviles con dimensiones pequeñas, cuyos SIM no van a ser intercambiados frecuentemente, se usa los SIMs del tipo plug-in en el formato ID-000. Sin embargo, la única diferencia entre los dos tipos de SIMs es el tamaño físico de las tarjetas. Sus características lógicas y físicas completamente idénticas.

[2] Referencia: [WOLFGANG]

5.3 TARJETAS INTELIGENTES SIN CONTACTOS PARA VIAJES AÉREOS

Este tipo de tarjetas ofrece a los clientes una amplia variedad de aplicaciones. Por ejemplo, éste es el sistema libre de tickets de la aerolínea alemana Lufthansa. Y se basa en memory cards sin contactos. El concepto básico de este tipo del sistema difiere de todas las otras aplicaciones de TIs descritas aquí, ya que las tarjetas se usan solamente para la identificación de los usuarios, con todos los datos de aplicación mantenidos en un sistema de segundo plano. La siguiente descripción relata la versión completamente equipada de la tarjeta. Naturalmente, existen versiones más simples, tales como las tarjetas que sólo tienen las funciones de las memory cards sin contactos. Con tales tarjetas, los viajeros pueden chequearse en un kiosco de servicio. Esto conduce naturalmente a un rendimiento más alto, con tiempos de espera más cortos y/o procesamiento más rápido. El sistema de bonus de Lufthansa también está integrado en la versión de la tarjeta de memoria sin contactos, haciendo innecesario usar alguna otra tarjeta o ingresar información adicional.

Usando el equipo automatizado adecuado colocado en el aeropuerto, un viajero puede registrarse y recibir la forma impresa de la 'información de abordaje'. Esta forma contiene todo la información esencial referente al vuelo registrado, similar a lo que está en un ticket actual. Si es necesario, se puede hacer una selección de butaca para el vuelo registrado mediante el kiosco. Los vuelos también pueden ser registrados por teléfono usando el número impreso en relieve en la tarjeta. En el futuro, Lufthansa planea administrar todo esto con tickets de papel, dado que toda la información pertinente puede ser recuperada del sistema en segundo plano mediante la tarjeta. Naturalmente, esto no deja sin lugar el hecho de poder hacer reservaciones o registro de vuelo mediante llamadas por teléfono o fax, lo cual aun será posible.

Además de estas funciones, la tarjeta también puede servir como una tarjeta de crédito por la incorporación de un holograma y una banda magnética con los datos necesarios. También es posible incrustar un chip con contactos en la ubicación ISO. Actualmente esto se usa para implementar un chip de tarjeta telefónica.

Todas las aplicaciones basadas en el chip sin contactos están estructuradas de forma muy simple con respecto a las TIs. La tarjeta sólo se usa para identificación, con el chip de memoria siendo usado para permitir autenticación dinámica por el sistema en segundo plano. Después de la autenticación, toda información almacenada en la tarjeta es leída. Actualmente, esto consiste del número del cliente, nombre del cliente y perfil del cliente. Con esta información, el sistema en segundo plano hace coincidir la tarjeta con una persona particular, después de lo cual los datos de registro, puntos de sistema de bonus y todas las otras funciones están disponibles. De este modo la tarjeta solo se usa como un tipo de llave, con los datos y mecanismos pertenecientes a las varias aplicaciones restantes en el sistema en segundo plano. Esto tiene ventajas considerables en este caso, dado que el sistema en segundo plano y toda su base de datos requerida, programas e interfaces ya están bien definidos.

Otra ventaja de este sistema es la forma en la que trata con las tarjetas extraviadas o defectuosas. Éste es un asunto difícil, el cual hasta ahora ha sido descuidado en otros sistemas que usan tarjetas multiaplicación. Si las aplicaciones han sido cargadas en la tarjeta después de la personalización, cuando el portador recibe una nueva tarjeta, el o ella deben contactar a todos los proveedores de las aplicaciones para recargarlas en su tarjeta. Gracias al sistema de arquitectura centralizada, la tarjeta inteligente de Lufthansa no tiene este problema. Si se pierde una tarjeta o se vuelve defectuosa, el cliente recibe una tarjeta nueva y la tarjeta antigua es bloqueada de lo ancho del sistema por medio de su número. Esto no requiere una gran cantidad de esfuerzo logístico, dado que todos los aeropuertos a los que sirve Lufthansa tienen acceso a los datos necesarios la red bien establecida de Lufthansa.

5.4 TARJETAS DE SEGURO MÉDICO

Para ilustrar esta aplicación de TIs, tomaremos como ejemplo la tarjeta de seguro médico en Alemania donde toda persona que cuente con un plan de seguro médico tiene una tarjeta (KVK). La tarjeta de miembro de seguro médico tiene dos funciones básicas para la persona asegurada. Su primera función es la de identificar ante el doctor a la persona que va a ser tratada por él. Esta reemplaza el papel de seguro médico. Su segunda función es actuar como un medio de almacenamiento de datos que pueden ser leídos por máquinas (PCs) en la clínica del doctor. Usualmente el terminal está conectado a la PC en la clínica, la cual también controla el terminal. La tarjeta puede ser leída usando el terminal y los datos de facturación que se obtienen de este modo pueden ser procesados automáticamente. Si el médico administra su práctica usando métodos tradicionales (esto es, sin PCs), el terminal puede directamente transferir los datos de la tarjeta a la impresora y entregar una forma impresa.

Tres entidades diferentes pueden acceder a la tarjeta de seguro médico. La primera es la clínica del doctor. Donde los datos solo pueden ser leídos de la tarjèta. Aquí no existe la intención de permitir la escritura de datos en la tarjeta y el software del terminal previene tales accesos. La segunda entidad es la organización de seguros médicos, la cual también solo puede leer los datos de la tarjeta. Aquí organización de seguros médicos también tiene terminales especiales que permiten la escritura de datos en la tarjeta. Esto puede ser necesario por ejemplo si la persona asegurada se muda a una nueva dirección. Sin embargo, muchos aseguradores simplemente emiten una nueva tarjeta a la persona asegurada en lugar de modificar los datos de la tarjeta ya existente y solicitan al portador de la tarjeta que destruya la antigua. Esto es mucho más simple y por tanto menos costoso.

Cuando la configuración de los datos fue especificada, fue muy importante para esto hacer posible que futuras extensiones o modificaciones pudieran ser

creadas sin ningún problema de compatibilidad. Consecuentemente, todos los datos personales en la tarjeta de seguro médico están estructurados usando el lenguaje de encripción ANS.1. Y son almacenados en la memoria de la tarjeta en una estructura TLV. Esto hace posible añadir otros elementos de datos en el futuro o cambiar los códigos usados para los objetos de datos existentes.

La tarjeta de seguro médico no es una tarjeta con microcontrolador. Es una memory card, con hardware muy similar al de las ampliamente usadas tarjetas telefónicas. La EEPROM usada debe tener una capacidad de al menos 256 bytes. Esto es igual a la cantidad de todos los datos necesarios ubicados en la tarjeta de seguro médico. Si la EEPROM es exactamente de este tamaño, los datos necesarios calzan justo y es físicamente imposible escribir algún dato adicional en la tarjeta en violación a la legislación de privacidad de datos.

Si el terminal está conectado a una computadora usando el protocolo de transmisión T=1 como especifica el estándar ISO/IEC 7816-3. Existe una restricción al respecto, la cual es que los datos encadenados no pueden ser usados en este protocolo. Hacerlo podría no añadir ninguna funcionalidad a la aplicación, pero podría incrementar la cantidad de memoria necesaria en el terminal. Dicho sea de paso, éste es un típico ejemplo del hecho que las aplicaciones reales a menudo usan solo las partes necesarias de estándares y no es común que todas las funciones especificadas en el estándar estén actualmente implementadas.

Existen solo tres comandos que el terminal puede ejecutar. El primer comando es un reset de la tarjeta de seguro médico, seguido por la recepción o lectura de la ATR. Este comando siempre es usado al iniciar una sesión para activar una tarjeta de seguro médico. El segundo comando es READ BINARY con el código ISO, el cual puede usarse para leer la porción de datos o todos los datos por medio del terminal. El tercer comando es WRITE BINARY, también en concordancia con el estándar ISO/IEC 7816-4, Sin embargo este comando está

solamente disponible en terminales administrativos. Éste es bloqueado en todos los otros tipos de terminales. La tarjeta de seguro médico de los aseguradores tiene protección de escritura implementada usando PINs que solamente son conocidos por la organización aseguradora. Las tarjetas para los planes de seguros médicos pueden ser escritas libremente si los comandos necesarios son conocidos.

Si el terminal es directamente conectado a la computadora, su función esencial es solo proveer una conversión entre el protocolo T=1 y el protocolo sincrónico dependiente del hardware de la tarjeta de seguro médico. No obstante, puede ser visto claramente que sería posible conmutar a tarjetas con microprocesador sin esfuerzo o costo indebido. Con las tarjetas con microprocesadores, la función única del terminal podría ser para relevar transparentemente los comandos recibidos de la computadora clínica. La respuesta de la tarjeta podría también ser retornada transparentemente a la computadora de control, sin ningún procesamiento del terminal.

5.5 SISTEMAS DE PEAJE ELECTRÓNICO

En algunos países, es común el cobro de peajes por el uso de ciertos caminos. En contraste con los cargos de tasa fija recolectados por la venta de etiquetas adhesivas, los peajes son recolectados de acuerdo al uso. En otras palabras, la cantidad recolectada depende de la frecuencia de uso y del tipo de vehículo. Hasta ahora, los peajes han sido pagados en efectivo en las casetas de peaje.

Un nuevo sistema para el cobro de peajes está siendo probado en Europa, particularmente en Alemania. Las pruebas de este tipo de aplicaciones deben satisfacer algunos de requerimientos. Primero el tráfico debería permitir un flujo normal y antecedentes inalterables de los puntos de recolección del peaje. Por

consiguiente, no debería lograrse evadir cobranzas de peaje conduciendo muy rápido. Además, las casetas de peaje o canastas de recolección no son permitidas para el uso de pago de peajes ya que la idea es instalar el equipo adecuado para montarlo en pasos a desnivel y puentes. Además, el sistema debe soportar tanto vías de un solo carril como vías múltiples.

Todos los sistemas probados tienen un dispositivo montado en el vehículo llamado ya sea OBU (onboard unit) o IVU (in- vehicle unit) y equipo adicional si es necesario. Este equipo obtiene energía de la batería del vehículo. En el compartimiento del pasajero cada sistema tiene un terminal de TIs, un display y un teclado simple. Existe un enlace con el mundo exterior, el cual podría ser unidireccional o bidireccional, dependiendo del sistema. Éste es esencialmente un enlace de microonda en la banda de frecuencia de 5,795 – 5,805 GHz. Alternativamente, algunos sistemas señales de radio en el rango de 40 – 500 MHz o luz infrarroja. La desventaja de usar luz infrarroja es naturalmente que la transmisión es fuertemente afectada por las condiciones ambientales, tales como una fuerte nevada o bruma.

Las TIs no contienen monederos electrónicos sofisticados, sino solamente comandos de débito simples y rápidos. Esto sucede en parte debido a que en las pruebas de este sistema no se usa dinero real y en parte debido al corto tiempo disponible para las transacciones. En algunos sistemas, los procesos de optimización van más lejos hasta a reducir la ATR a cuatro bytes, para dejar el tiempo suficiente para el débito que debe realizarse. Esto es entendible en vista de los requerimientos de un flujo de tráfico no restringido. A 250 km/h, un vehículo cubre 70 m por segundo. La estación de control tiene un rango de comunicaciones de aproximadamente 5 m en la banda de frecuencia de 5,8 GHz. Esto arroja un tiempo de mora (tiempo en el cual el vehículo está en el rango de la estación) de alrededor de 70 ms. La tarjeta debe realizar los siguientes procesos en ese intervalo:

- Reset de la tarjeta y transmisión de la ATR (≈ 10 ms)
- Encripción DES para autenticar la tarjeta (≈ 12 ms)
- Acceso de escritura a la EEPROM para almacenar el nuevo balance ($\approx 2x3{,}5$ ms)
- Transmisión de datos desde y hacia el terminal (≈ 30 ms)

Es necesario tiempo adicional para la transmisión de datos entre el OBU y la estación de peaje. Las TIs operan a la máxima frecuencia de reloj disponible (usualmente 5 MHz). La tasa de transmisión de datos entre el OBU y la estación de peaje, a 1 Mbit/s, fue sustancialmente más alta que entre el OBU y la tarjeta. Tres tipos de sistemas fueron probados para este tipo de aplicaciones.

Sistema 1

En el primer sistema, el equipo necesario está instalado en dos pasos a desnivel o puentes sucesivos. Naturalmente el sistema está diseñado para que la recolección de peaje no sea afectada por cambio de vía entre dos estaciones. Existen dos tipos de pago diferentes. En el modo pospago, facturas de peaje acumuladas son recolectadas de una cuenta bancaria, como con una tarjeta de crédito. En el modo prepago, se usan TIs prepagadas y la cantidad apropiada es deducida del balance en la tarjeta cada vez que el peaje es recolectado.

Sistema 2

El segundo sistema no está basado en una infraestructura en las calles. En lugar de eso, usa estaciones de peaje virtuales, las cuales existen solo en la base de datos del OBU del vehículo. Los OBUs contienen las coordenadas de todas las estaciones de peaje disponibles, junto con las tasas de peaje para los segmentos correspondientes de la carretera. Tan pronto como un vehículo ingresa a un segmento de carretera definido, la cantidad de peaje es debitada de la TI.

Naturalmente, este proceso es menos dependiente del tiempo que el primer sistema.

Sistema 3

El tercer sistema representa la solución más técnica para la recolección automática de peaje de vehículos. El OBU está equipado con un receptor GPS y un teléfono móvil GSM. La posición del vehículo es determinada usando la unidad GPS y el teléfono móvil se usa para obtener los datos de precio específico. Como con los otros dos sistemas, el OBU contiene un terminal con una TI que se usa para la recolección del peaje.

5.6 FIRMAS DIGITALES

Existen dos prerrequisitos fundamentales para el uso de firmas digitales. La primera es una TI con microcontrolador con un poderosos coprocesador numérico y la segunda es la existencia de una estructura legal general claramente definida. La TI es usada para el almacenamiento seguro de las llaves de firma secretas y para generar las firmas digitales. La legislación establece las condiciones obligatorias que se aplican a todas las partes en esta aplicación de las TIs.

Por supuesto, las firmas digitales también pueden usarse en la forma de una aplicación cerrada que es completamente independiente de cualquier contexto legal. Sin embargo, en este caso planes contractuales entre las partes participantes son necesarias para hacer que una firma digital actúe sobre la persona que genera la firma. En este sector negocio-a-negocio, por ejemplo, el uso de firmas digitales ha sido una práctica estándar por varios años. Sin embargo, si el objetivo es un sistema abierto que puede ser usado por partes que no se conocen unas con otras, son necesarias condiciones legales adecuadas. Esta estructura legal permite que las firmas digitales sean equivalentes a las

firmas normales y hace posible entablar juicios en los tribunales con respecto a documentos firmados de esta forma, al igual que si fuesen firmados a mano.

Las aplicaciones posibles para las firmas digitales reconocidas legalmente, las cuales de esta forma son equivalentes a las firmas a mano, son prácticamente ilimitadas. En el caso más simple, pueden usarse para firmar electrónicamente cartas u órdenes, pero también pueden utilizarse en un sistema de pago electrónico para firmar autorizaciones de débito directo o en sistema bancario a domicilio para firmar órdenes de remisión. La aplicación más importante para las firmas digitales será indudablemente la de firmar todo tipo de arreglos contractuales.

Los dos componentes esenciales de un sistema de firmas digitales son el centro de confianza y la tarjeta de firmas.

Es necesario un ambiente de hardware seguro para almacenar y usar las llaves privadas de firmas. Las TIs representan una solución ideal para este requerimiento, dado que ellas son físicamente pequeñas, no son caras y ofrecen un alto nivel de protección contra lecturas o modificaciones externas de los datos almacenados.

Los algoritmos de criptografía usados para este propósito requieren un microcontrolador con una unidad de procesamiento numérico suplementaria (NPU) para permitir que las firmas digitales sean generadas y probadas en un periodo de tiempo aceptable. En teoría, un documento firmado también podría ser probado en el exterior de la tarjeta de firmas, dado que toda la información necesaria también está disponible allí. Si esto se vuelve significativo en la práctica está por verse. En términos de seguridad, sin embargo, es mejor realizar las pruebas dentro de la tarjeta.

Las llaves pública y privada para las firmas digitales pueden ser generadas en una manera ya sea centralizada o descentralizada. En cualquier caso, las llaves deben ser generadas en un ambiente criptográficamente seguro, tal como dentro de un módulo de seguridad o TI. También es importante para la llave pública que luego sea firmada usando la llave privada del centro de confianza. Es necesario asegurar que las llaves pública y privada de la tarjeta de firmas sean reconocidas como auténtica por el centro de confianza. Sin este reconocimiento, no sería posible que una tercera parte verifique que la firma de un documento electrónico en realidad proviene de un emisor genuino de un sistema de firmas digitales.

5.7 OTRAS APLICACIONES

Si bien las aplicaciones antes mencionadas cubren la mayor parte de aplicaciones posibles para las TIs, como también se dijo antes dichas áreas de aplicación son muy extensas y solo basta con tener un tanto de imaginación para integrar una TI en casi cualquier ámbito de la actividad humana, a continuación se muestran algunas otras áreas donde también las TIs están siendo usadas exitosamente y un resumen de las ya mencionadas:

MONEDEROS ELECTRÓNICOS: Estas tarjetas electrónicas de pago, generalmente emitidas por un banco, permiten a los portadores de tarjetas evitar los problemas de encontrar el cambio exacto al cargar efectivo de sus cuentas bancarias en un monedero electrónico que puede usarse para las compras menudas de todos los días en tiendas, kioscos, máquina automáticas, expendedoras de boletos de transporte, parquímetros, cabinas telefónicas, etc.

CONTROL DE SEGURIDAD DE ACCESO: Las TIs pueden programarse para permitir el acceso a edificios o datos, dependiendo del cargo y del nivel de autorización.

TARJETAS TELÉFONICAS: Las TIs pueden identificar a la persona que llama desde un teléfono público y cargar el coste de la llamada a una cuenta de teléfono.

TARJETA DE ASISTENCIA MÉDICA: Las TIs pueden almacenar expedientes médicos, información para casos de emergencia y situación en materia de seguro de enfermedad.

CREDENCIALES DEL GOBIERNO: Los gobiernos que deseen reducir costes y papeleo pueden utilizar TIs para emitir, licencias de conducir, pasaportes y visas.

TARJETAS DEL SEGURO SOCIAL: Las TIs pueden almacenar prestaciones, pago de pensiones y reembolsos del seguro de enfermedad en un monedero electrónico y pueden identificar con toda seguridad a los portadores de tarjetas que solicitan prestaciones.

TARJETAS URBANAS / UNIVERSITARIAS / DE EMPRESA / DE CLUBES DEPORTIVOS: Las TIs están muy extendidas entre los grupos cerrados de usuarios (residentes de una ciudad, personal de una universidad, personal de una compañía, aficionados de un equipo deportivo, clientes de un parque de atracciones, etc.), quienes pueden utilizar tarjetas con múltiples aplicaciones para pagar sus cuotas o acceder a servicios (por ejemplo, comidas en la cafetería, máquinas expendedoras de bebidas, credenciales de biblioteca, entradas para centros de ocio, estadios o parques de atracciones, estacionamientos, subidas a juegos mecánicos, etc.), de forma diferenciada de acuerdo con la autorización y circunstancias personales.

PAGOS SEGUROS POR INTERNET: Las TIs ofrecen un medio seguro para pagar por Internet que los consumidores exigen a medida que crece el comercio electrónico. La adopción de normas comunes por parte de los emisores de

tarjetas permitirá una interoperabilidad internacional de las tarjetas, de forma que los pagos podrán hacerse a través de las fronteras nacionales.

COBRO AUTOMÁTICO DE TARIFAS DE TRANSPORTES PÚBLICOS: Las compañías de transporte público han adoptado TIs sin contacto en muchas grandes zonas metropolitanas. A estas tarjetas se les puede añadir la tecnología de las TIs para identificar a los portadores de tarjetas que tiene derecho a tarifas con descuento o para permitir la compra de más boletos electrónicos con un monedero electrónico.

TARJETAS DE FIDELIDAD PARA CLIENTES: En varios programas de fomento de fidelidad de los clientes de líneas aéreas, hoteles, restaurantes de comida rápida y grandes almacenes se utilizan TIs que registran los puntos y premios y ofrecen datos detallados sobre los hábitos de los clientes a los operadores de dichos programas, a fin de elaborar sus campañas de promoción con mayor precisión.

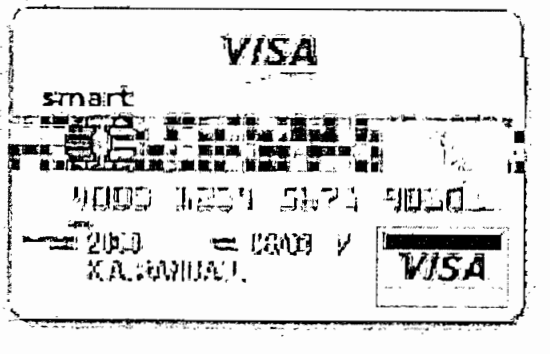

Figura 5.2 Tarjetas inteligentes de crédito. (*Referencia:* [INGENICO])

TARJETAS DE CRÉDITO / DÉBITO: En varios países, las versiones para cinta magnética de estas aplicaciones residen conjuntamente en las TIs, pero las nuevas normas EMV implican que los adquiridores deben actualizar sus redes,

Capítulo 6

Diseño de un ejemplo de aplicación

6.1 OBJETIVOS

El caso de estudio de este capítulo se centra en desarrollar un prototipo de un sistema que permita ilustrar el uso de una tarjeta inteligente en el ambiente estudiantil, esto con el fin de poner en práctica los conocimientos obtenidos y evaluar las herramientas utilizadas para el desarrollo.

Los dos objetivos principales son:

- Determinar la aplicación que permita poner en práctica algunas de las principales características que presentan las tarjetas inteligentes, como son su capacidad de almacenamiento, ciertas propiedades de seguridad así como también la facilidad de usarlas.

- Construir un interfaz gráfico (software) que permita almacenar los datos en la tarjeta inteligente y leer su contenido.

6.2 APLICACIÓN PROPUESTA

La aplicación propuesta consiste en almacenar en la tarjeta el curriculum académico de los estudiantes de la carrera de Ingeniería Electrónica y Telecomunicaciones, en este caso en particular la implementación va a ser desarrollada para una Memory Card con capacidad de 1024 bytes, dado que como se dijo en un principio el objetivo de este trabajo es ilustrar el uso de tarjetas

inteligentes en el ambiente estudiantil, para lo cual es suficiente usar el tipo de tarjetas antes mencionado. Este tipo de aplicación permitirá contar con una credencial académica que no solo permita autenticar la identidad de un estudiante para permitir su acceso a ciertos sitios o a cierta información, sino conocer su desempeño académico como una variable a ser tomada en cuenta en algunas ocasiones.

6.2.1 ANTECEDENTES

En los capítulos anteriores se realizó un estudio de la tecnología de las tarjetas inteligentes en los que se analizaron puntos clave como su capacidad de almacenamiento (que es mayor que la de una tarjeta de banda magnética). También como principal ventaja se anotó la facilidad de su manejo, así como ciertas características de seguridad con las que cuentan, partiendo de esto se sabe que dichas tarjetas están siendo utilizadas en instituciones educativas con mucho éxito, en términos generales se utilizan como dinero electrónico, medios de acceso, registro de libros prestados en bibliotecas, etc. Aquí se diseñará algo diferente a las aplicaciones antes expuestas, pero no por esto se dejan de lado los puntos clave antes mencionados, como son la capacidad de almacenamiento y la facilidad de uso.

6.2.2 REQUERIMIENTOS

Al ser el curriculum académico un documento muy familiar para nosotros los estudiantes, no fue muy complejo comprender la estructura básica del mismo y su contenido. Partiendo del documento en formato de texto se determinó que se almacenaría en la tarjeta la misma información que se presentaba en dicho documento.

Al usar una Memory card cuya capacidad máxima es de 1024 bytes, es necesario optimizar al máximo el espacio que será utilizado en la memoria, por tanto se debe definir una forma de codificación de los datos y cómo almacenarlos eficientemente.

La información de interés sobre cada estudiante está comprendida en las siguientes categorías:

- Datos administrativos

- Datos académicos

6.2.3 CARACTERÍSTICAS DEL PROBLEMA

Al realizar un análisis más profundo de la estructura y características de la información que maneja el problema, se observó lo siguiente:

- La estructura de la información que va a ser almacenada es conocida y consta de partes estáticas (por ejemplo el nombre del estudiante) y partes dinámicas (como son los registros de materias tomadas).

- Esta información ya se encuentra disponible en las bases de datos de la Carrera, por lo cual simplemente se debería interactuar con dicha base de datos para actualizar la información contenida en la tarjeta.

- El número de materias tomadas por el estudiante en el transcurso de su vida académica dependerá de si el estudiante aprobó o no dicha materia cuando la tomó por primera vez, caso contrario es conocido que el estudiante podrá

intentar aprobar dicha materia por dos ocasiones más como máximo. Es decir que un estudiante podría terminar su carrera en el peor de los casos tomado cada materia un máximo de 3 veces, lo que arrojaría un número de registros de materia aproximadamente igual a 210, pero en la práctica ningún estudiante tendrá entre sus planes terminar la carrera en aproximadamente 15 años que es lo que implicaría realizar lo antes expuesto. Por esta razón se ha determinado que el 66% del valor anterior de registros sería un valor aceptable. Lo que quiere decir 140 registros.

- Por otro lado tenemos que los datos de los registros son en su mayoría datos numéricos, como: nota, número de matricula, período, año, créditos.

- Dado que se cuenta con los valores de espacios requeridos en memoria máximos y mínimos para los diferentes tipos de datos, es posible realizar una división fija de memoria. Es decir se contará con un mapa de memoria bien definido.

- Con la finalidad de almacenar toda la información antes mencionada se realizará la codificación de la misma. Además se contará con un sistema de comprobación de errores, que permitirá evaluar si la información almacenada en la tarjeta no ha sido adulterada.

- Dado que la información almacenada en la tarjeta está codificada, el proceso de lectura será básicamente de decodificación y verificación de la veracidad de la información contenida en la tarjeta. Esto implica contar con una aplicación muy sencilla que sea capaz de mostrar un mensaje de error en caso de que los datos hayan sido alterados, de otro modo deberá mostrar el contenido de la tarjeta, obviamente decodificado, por medio de un interfaz visual.

6.2.4 HERRAMIENTAS UTILIZADAS EN LA SOLUCIÓN

Para realizar la implantación de la aplicación se utilizarán principalmente las siguientes herramientas:

6.2.4.1 Lector y Programador

El lector y programador que se utilizó es un producto de ACS llamado ACR30 reader. Utiliza el puerto serial para la transferencia de datos y un puerto PS/2 para el suministro de energía, pudiéndose conectar el mouse o el teclado en el mismo puerto PS/2. Se comunica con la Smart Card a través de software propietario embebido en el lector.

6.2.4.2 Kit

Para el desarrollo se utilizó un kit de desarrollo de ACS, que incluye el lector descrito anteriormente y 5 Memory cards SLE4428. La razón de usar este Kit es que su precio es conveniente y reúne las características necesarias para el trabajo que se va a realizar.

El kit, llamado AC-Kit Smart Card Development Pack, provee las siguientes facilidades:

- Interacción con la tarjeta a nivel de APDU
- Documentación del API que se usa para el intercambio de comandos y respuestas.
- Una librería que permite que la escritura y lectura de la tarjeta se realice de manera muy simple.

6.2.4.3 Memory Card SLE4428

Como su nombre lo indica ésta es una Memory Card y cuenta con una EEPROM de 1KByte, posee también protección de escritura y seguridad lógica.

6.3 IMPLEMENTACIÓN DE LA APLICACIÓN

Como se anotó antes entre los objetivos principales de este capítulo se tiene la implementación de un interfaz visual que permita cargar la tarjeta con los datos correspondientes al curriculum académico, a esta etapa del proyecto se la llamará escritura de la tarjeta, la segunda etapa consiste en mostrar el curriculum almacenado en la tarjeta, esto también mediante un interfaz visual (etapa de lectura).

Para llevar a cabo la solución propuesta, en la etapa de escritura se debe contar con una aplicación que permita realizar la conversión de los distintos datos en bytes, esto significa contar con tablas que relacionen números entre 0 y 254 con cadenas de caracteres o números decimales según sea el caso. Mientras que en la etapa de lectura se requiere interactuar también con tablas. Esta solución es factible en términos del espacio de memoria necesario y disponible en la Memory Card y procesos necesarios para desarrollar las aplicaciones.

6.3.1 MAPA DE MEMORIA

La memoria ha sido dividida como se muestra en la figura 6.1 Allí se observa que los primeros 27 Bytes son de solo lectura y en ellos se encuentran escritos datos del fabricante.

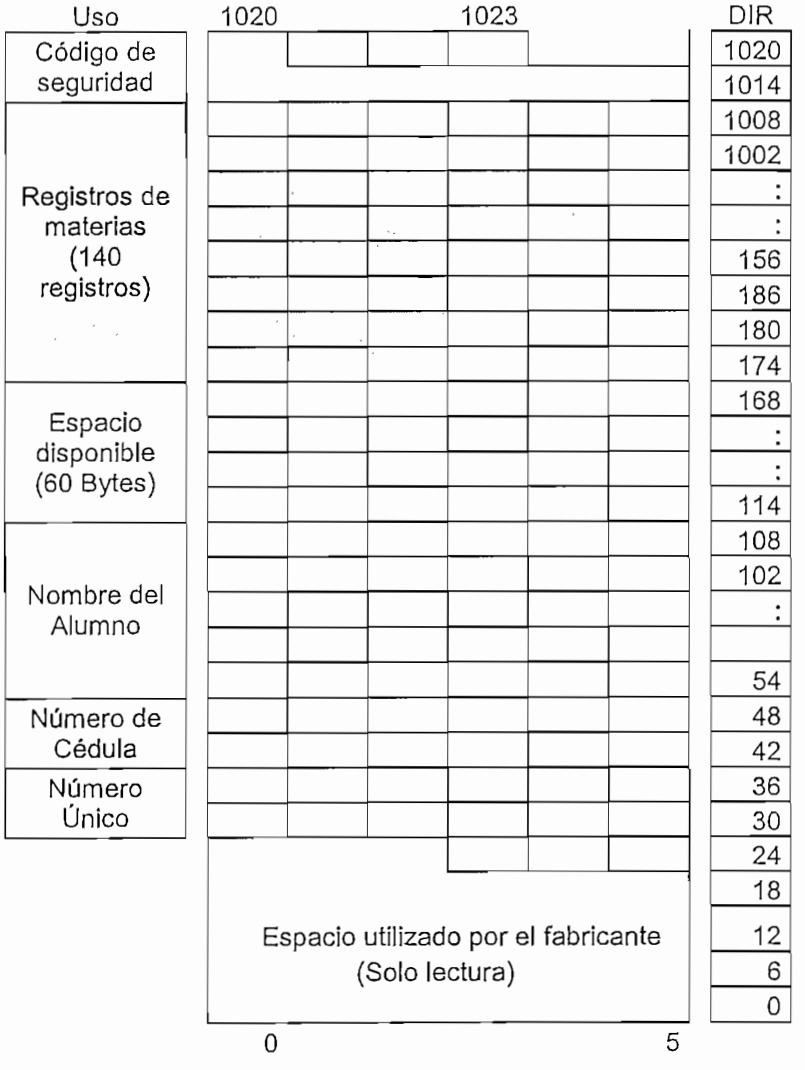

Figura 6.1 División de la memoria EEPROM de la Memory Card

A partir del Byte ubicado en la dirección 27 (H1B) se escriben 11 Bytes correspondientes a los caracteres que forman la cadena del número único. El Byte número 12 es el resultado de una operación XOR entre los 11 anteriores y se usa para realizar detección de errores. A partir del Byte ubicado en la dirección 39 (H27) se escriben 12 Bytes correspondientes a los caracteres que forman la cadena del número de cédula. El Byte número 13 es el resultado de una operación XOR entre los 12 anteriores y se usa para realizar detección de errores.

A partir del Byte ubicado en la dirección 52 (H34) se escriben 60 Bytes correspondientes a los caracteres que forman la cadena del nombre del alumno. Y el Byte número 61 es el resultado de una operación XOR entre los 60 anteriores y se usa para realizar detección de errores.

Byte 1
Año
255 Años

Byte 2											
b8	bit 9 de nota	b7	b6	b5	bits no utilizados	b4	b3	bit 10 y bit 9 de materia	b2	b1	Matrícula
0	0					0	0	0	0	0	1
1	1					0	1	1	0	1	2
						1	0	2	1	0	3
						1	1	3	1	1	4

Byte 4
8 bits de Nota
Byte 4 + 1 bit del Byte 2 Nota Máxima= 51,2

Byte 3
8 bits de Materia
Byte 3 + 2 bits del Byte 2 1024 Materias

Byte 5										
b8	b7	Aprobación	b6	b5	Período	b4	b3	b2	b1	Créditos
0	0	E	0	0	1	0	0	0	0	0
0	1	A	0	1	2	0	0	0	1	1
1	0	F	1	0	3	0	0	1	0	2
1	1	X	1	1	4	0	0	1	1	3
						0	1	0	0	4
						0	1	0	1	5
						0	1	1	0	6
						0	1	1	1	7
						1	0	0	0	8
						1	0	0	1	9
						1	0	1	0	10
						1	0	1	1	11
						1	1	0	0	12
						1	1	0	1	13
						1	1	1	0	14
						1	1	1	1	20

Figura 6.2 Secuencia de Bytes escritos para cada registro y codificación de los campos Año, Nota, Período, Tipo de aprobación, Número de matrícula, Créditos, Número de Materia.

A partir del Byte ubicado en la dirección 174 (HAE) se escriben los registros de cada materia utilizando para el efecto 6 Bytes por registro, en los primeros 5 bytes se encuentran codificados los datos de cada registro como se muestra en la figura 6.2 y en el Byte número 6 el resultado de una operación XOR entre los cinco anteriores.

Para almacenar el dato correspondiente al **año** se utiliza un Byte por registro lo cual permite tener un intervalo de 256 años, en este caso en particular dado que se cuentan con registros desde el año 1900 se podrán almacenar registros hasta el año 2156.

Para almacenar el dato correspondiente a la **materia** se utilizan 10 bits, lo cual permite codificar 1024 materias, esto es suficiente dado que la tabla de la base de datos que mantiene almacenada esta información cuenta con 857 materias. Los 10 bits se encuentran almacenados en la tarjeta como se muestra en la figura 6.2

Para almacenar el dato correspondiente con la **nota** que aparece en el registro se utilizan 9 bits, esto debido a que la nota máxima puede ser de 40 y las notas cuentan con un decimal. Al usar 9 bits podríamos llegar a tener una nota máxima de 51,2 lo cual para el caso es suficiente. Los 9 bits se encuentran almacenados en la tarjeta como se muestra en la figura 6.2.

Para almacenar el dato correspondiente al **período** que aparece en cada registro se utilizan 2 bits, lo cual permite codificar cuatro períodos (1, 2, 3, 4) aunque en la práctica solamente se cuentan con registros con los períodos 1, 2 y 3. Los 2 bits se encuentran almacenados en la tarjeta como se muestra en la figura 6.2.

Para almacenar el dato correspondiente al **tipo de aprobación** se utilizan 2 bits, lo cual permite codificar cuatro tipos de aprobación que son E (Exonerado), A (Aprobado), F (Fallido) y X (Exonerado certificado). Los dos bits se encuentran almacenados en la tarjeta como se muestra en la figura 6.2.

Para almacenar el dato correspondiente a los **créditos** se utilizan 4 bits, lo cual permite codificar materias con cero créditos hasta materias con 15 créditos, para nuestro caso existen materias cuyos créditos correspondientes van desde 0 hasta 8 y como caso excepcional está el proyecto de titulación el cual cuenta con 20 créditos. Los bits se encuentran almacenados en la tarjeta como se muestra en la figura 6.2. Además cave señalar que estos datos de los créditos se utilizan para comprobar la veracidad de los datos almacenados. Los 4 bits se encuentran almacenados en la tarjeta como se muestra en la figura 6.2.

Para almacenar el dato correspondiente al **número de matrícula** se cuenta con 2 bits, lo cual permite codificar hasta cuatro matrículas en una misma materia. Los 2 bits se encuentran almacenados en la tarjeta como se muestra en la figura 6.2.

Tabla 6.1 Tamaño del campo correspondiente a datos a ser almacenados.

Nombre del campo	Tipo de datos	Tamaño del campo
ANOMAT	Número	Doble
APRUEBM	Texto	1
CEDULA	Texto	12
CODEST	Texto	11
CODMAT	Texto	6
CREDMAT	Número	Doble
NOMEST	Texto	60
NOMMAT	Texto	60
NUMMAT	Número	Entero
NUMMTR	Número	Doble
SEMMAT	Número	Doble
SUMATOM	Número	Doble
TOTCRED	Número	Doble

Para dimensionar el tamaño de los datos en la tarjeta se utilizó el parámetro "Tamaño del campo" que provee la base datos en la que está almacenada la información académica de los estudiantes. Esta información se muestra en la tabla 6.1

6.3.2 ETAPA DE ESCRITURA

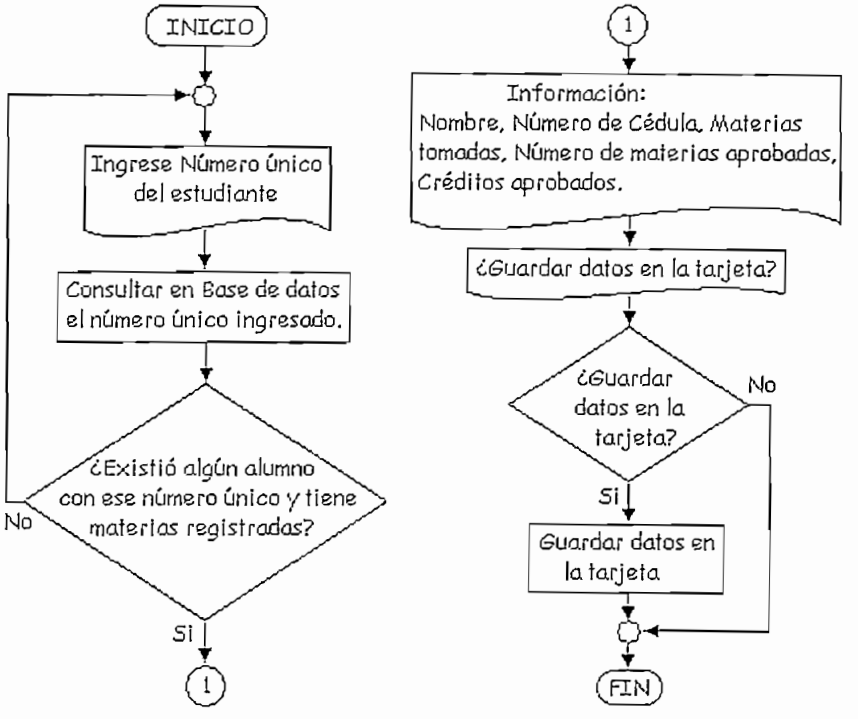

Figura 6.3 Diagrama de flujo de la aplicación de escritura de curriculum en la tarjeta.

En esta etapa como su nombre lo indica se realizará la implementación de una aplicación de software que permita cargar los datos del estudiante en la tarjeta y guardar también los respectivos Bytes de verificación de la información.

Para realizar esta operación se sigue el diagrama de flujo mostrado en la Figura 6.3. Como se puede ver el bloque que mas se relaciona con el tema de las tarjetas inteligentes es el de escribir los datos en la tarjeta. Por tanto profundizaremos mayormente en ese bloque. La escritura de los datos en la tarjeta se realiza siguiendo el diagrama de flujo mostrado en las Figuras 6.4 y 6.5.

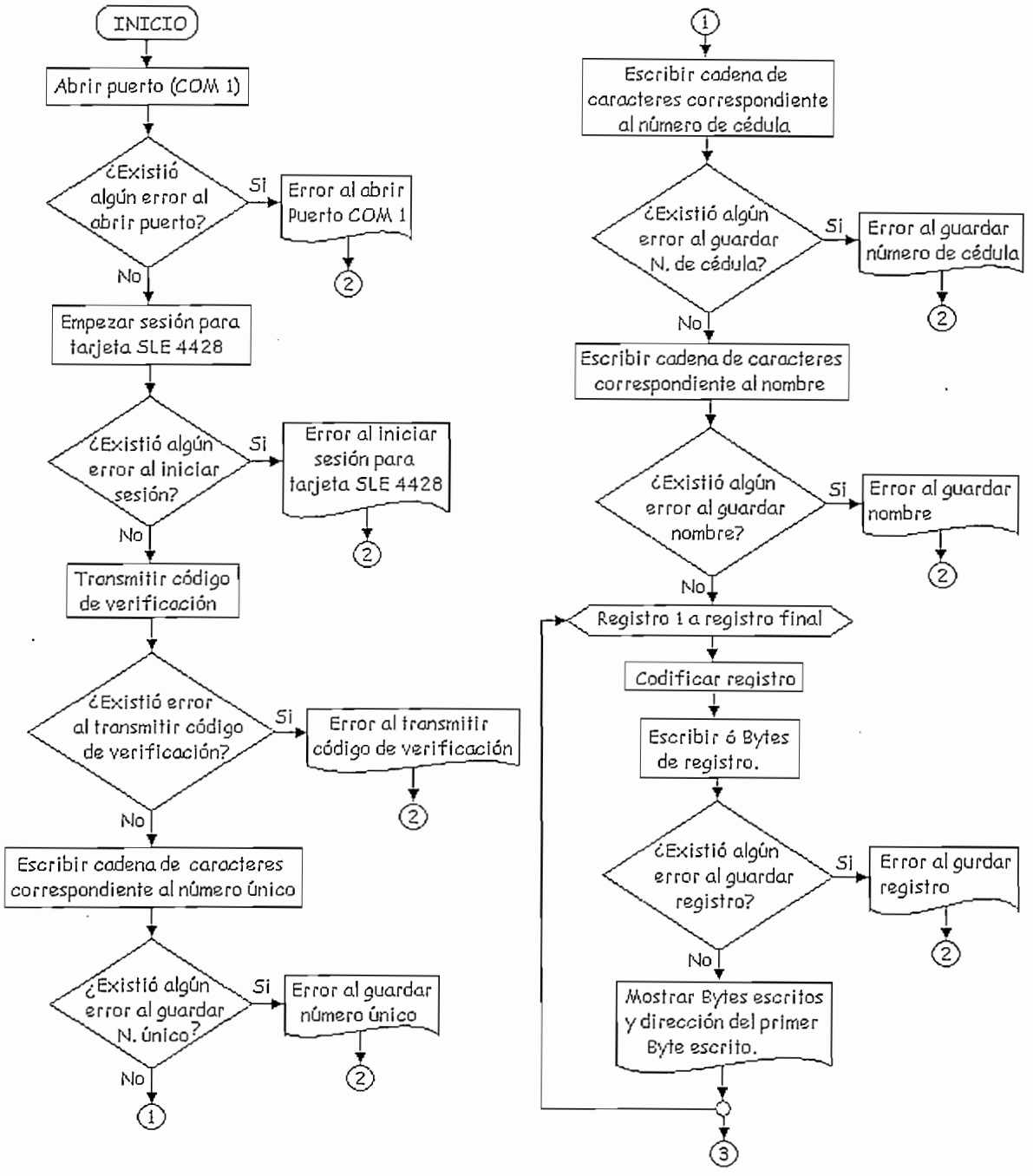

Figura 6.4 Diagrama de flujo del bloque Escribir en tarjeta.

Como se observa en la figura 6.3 solamente se tiene la posibilidad de escribir en la tarjeta una vez que se ingresa el número único de un estudiante de la carrera de Electrónica y Telecomunicaciones y se comprueba que dicho

estudiante tiene registros en la tabla SAEMATER que almacena las materias tomadas por un estudiante y las notas que obtuvo al tomar dichas materias.

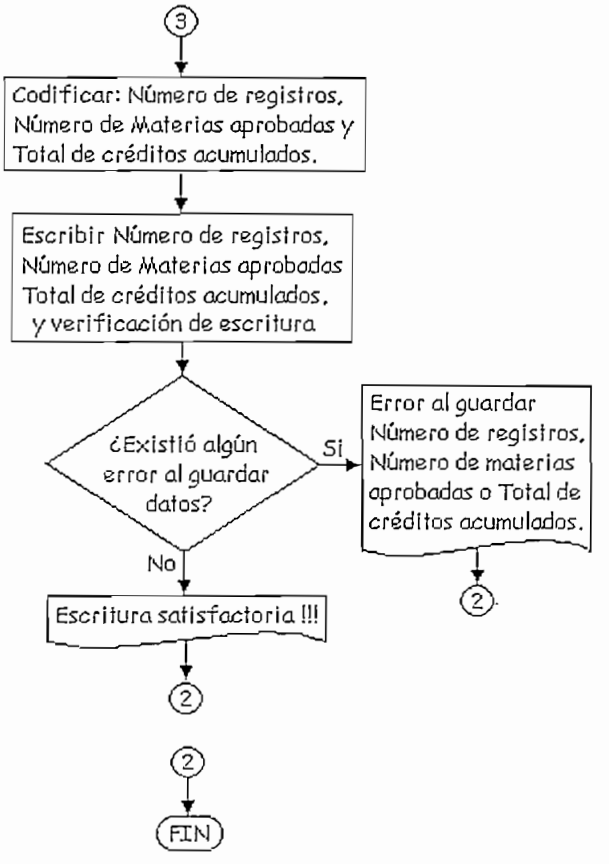

Figura 6.4 Diagrama de flujo del bloque Escribir en tarjeta (Continuación).

Una vez que un número único válido ha sido ingresado en el interfaz visual se muestran los datos del estudiante como se muestra en la figura 6.5. Caso contrario se muestra el mensaje de error de la figura 6.6.

En esta etapa se implementa como fue descrito anteriormente los diferentes mecanismos de control de errores, los cuales básicamente consisten en escribir un Byte que es el resultado de una operación XOR de los datos relacionados con dicho Byte. Más tarde, en la etapa de lectura este Byte será leído y se tendrá la posibilidad de generar un Byte realizando la misma operación con los datos relacionados con dicho byte, finalmente se comparan el Byte de

control de errores leído y el Byte de control de errores generado con el fin de observar si los datos fueron o no alterados.

ESCRITOR

Buscar: Buscar 9820905 | Grabar: Grabar | Salir: Salir

Nombre: CHIMBORAZO TORO ROBERTO LUIS | Cédula: 1803394145 | Créditos: 219 | Mat. Apr.: 52

CURRICULUM ACADÉMICO ESTUDIANTIL

Año	Per	Mat	Nota	Crd	Aprb	Código	Materia
1999	2	1	35	7	E	IGF117	ALGEBRA LINEAL
1999	2	1	33	7	A	IGF127	CALCULO
1999	2	1	30	8	A	IGF138	FISICA GENERAL I
1999	2	1	30	4	E	IGF144	QUIMICA GENERAL I
2000	1	1	31	6	E	IGF216	ECUACIONES DIFERENCIALES
2000	1	1	31	6	E	IGF226	CALCULO VECTORIAL
2000	1	1	30	8	E	IGF238	FISICA GENERAL II
2000	1	1	25	4	A	IGF244	QUIMICA GENERAL II
2000	1	1	35	2	E	IGF252	COMPUTACION

DISTRIBUCIÓN DE BYTES EN MEMORIA

Dirección <= Byte 1 Xor Byte 2 Xor Byte 3 Xor Byte 4 Xor Byte 5 = Byte 6

Figura 6.5 Pantalla del interfaz del escritor con un número único válido ingresado.

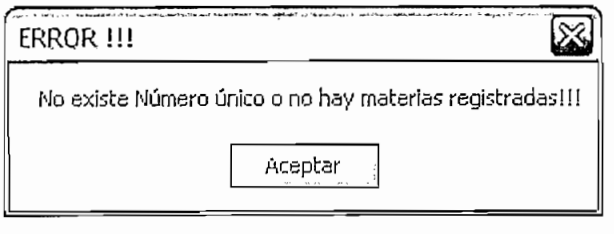

Figura 6.6 Mensaje de Error al ingresar número único no válido.

Al pulsar el botón Grabar los datos son transmitidos a la tarjeta, en el recuadro denominado Distribución de Bytes en Memoria se anota la dirección de inicio de los seis Bytes que corresponden a cada registro seguida del valor de los Bytes escritos. Al finalizar la escritura en la tarjeta se muestra un mensaje que confirma que la escritura fue exitosa.

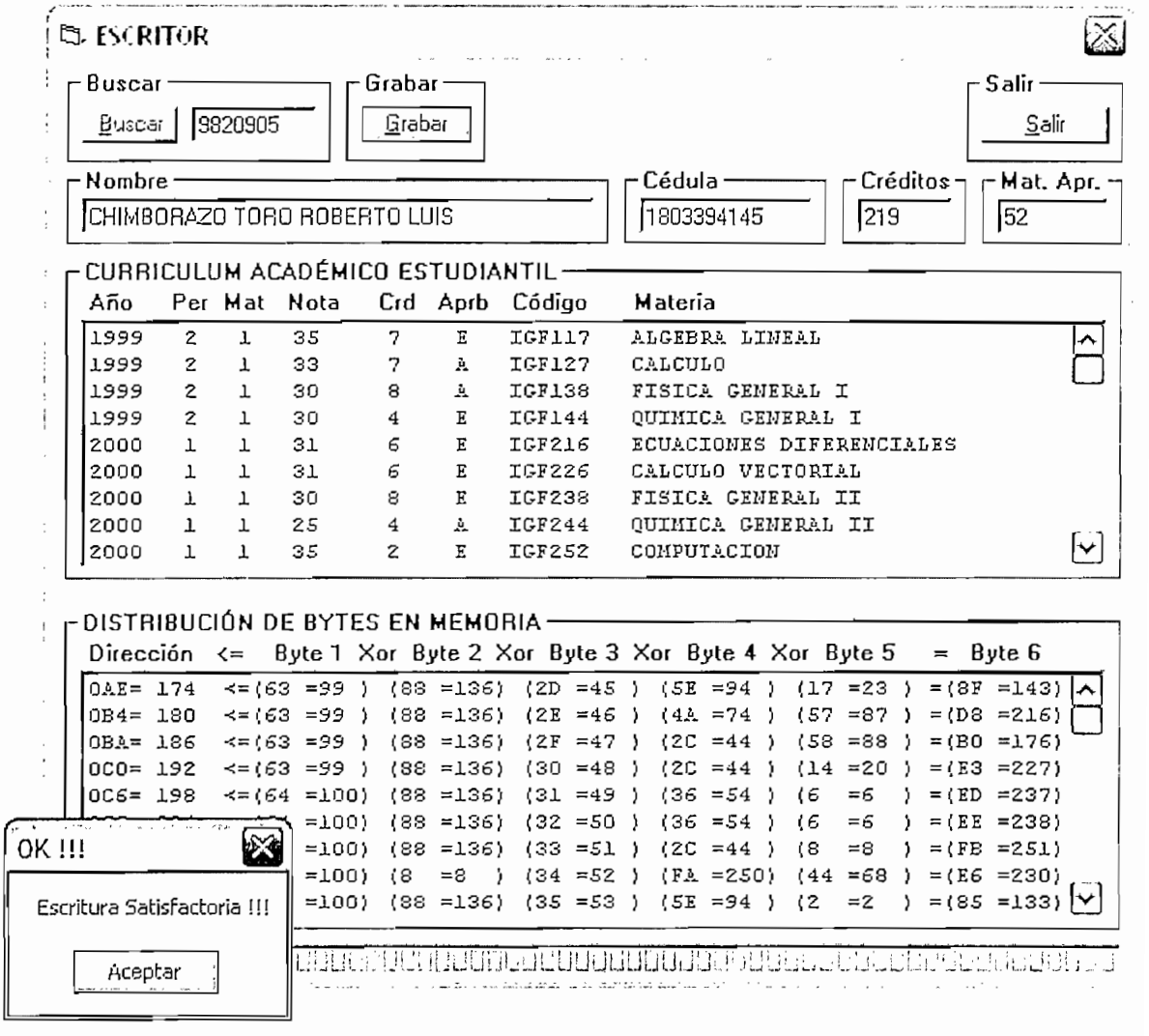

Figura 6.7 Ejemplo de escritura satisfactoria de curriculum en la tarjeta.

Para la implementación de esta aplicación se consultaron las siguientes tablas de la base de datos:

SAEESTUD contiene datos del estudiante como número único, número de cédula, créditos acumulados.

SAEMATER contiene todas las materias tomadas por el estudiante desde su ingreso a la carrera así como las notas y si las aprobó o no.

SAENOMAT contiene información referente a las materias que pueden ser tomadas por los alumnos de la carrera, así como el código de la materia propio del sistema y el código ingresado para la realización de esta aplicación.

6.3.3 ETAPA DE LECTURA

En esta etapa se realiza la lectura de los datos almacenados en la tarjeta, para el efecto se toma en cuenta el diagrama de flujo que se muestra en las figura 6.10.

Como se sabe los datos de Nombre, Número Único y Número de Cédula están almacenados en la tarjeta como una cadena de caracteres lo cual implica un Byte por cada caracter. Por tanto para leer estos datos no es necesario consultar la base de datos ya que éstos están escritos en su totalidad en la tarjeta.

Por otro lado para leer los registros se requiere realizar primero la decodificación de aquellos datos que están escritos en la tarjeta como son: nota, año, período, créditos, tipo de aprobación y código de la materia, este último no es el código que aparece en el curriculum sino un número entre 1 y 857 utilizado para optimizar la aplicación. Con este número se consulta la base de datos, más precisamente la tabla SAENOMAT para obtener el nombre de la materia y el código respectivo. De la misma forma se procede con cada registro.

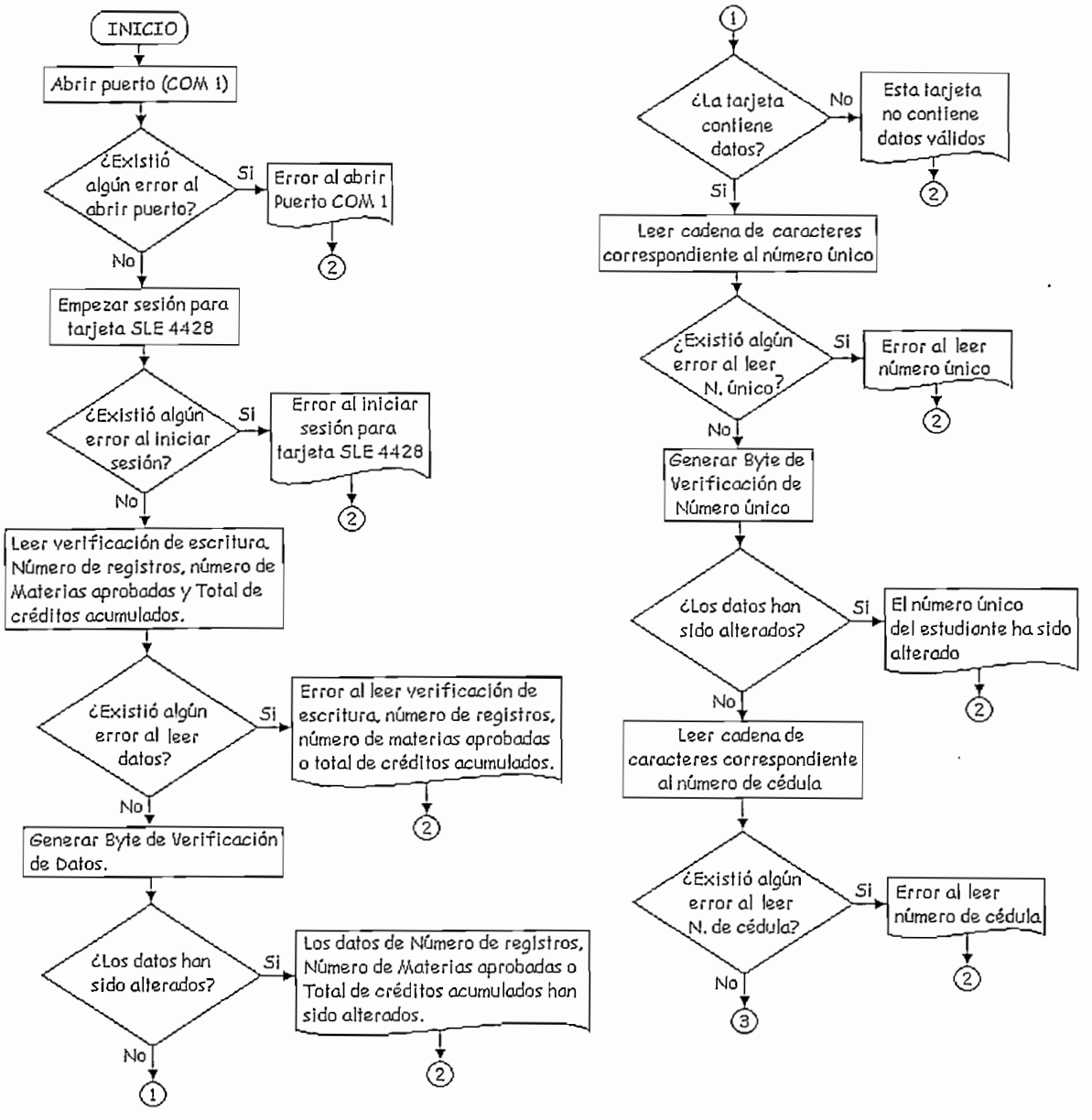

Figura 6.8 Diagrama de flujo de la aplicación de lectura de la tarjeta.

Cave anotar que para cada registro se realiza la comprobación de que los datos no hayan sido alterados, esto tomando en cuenta los Bytes de control de errores anteriormente mencionados y que fueron escritos como es lógico en la etapa de escritura. El interfaz de usuario para el proceso de lectura es el mostrado en la figura 6.11.

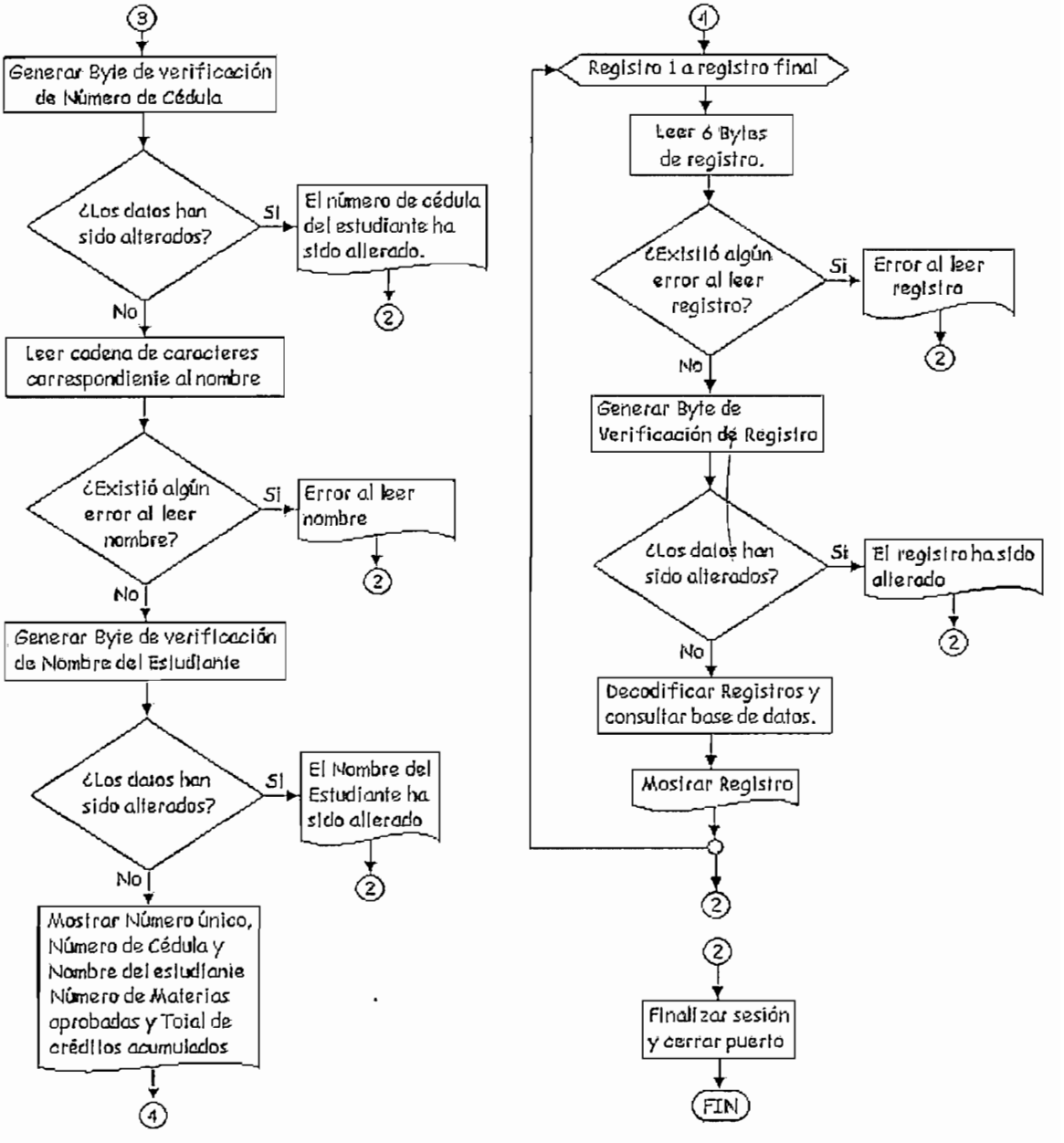

Figura 6.8 Diagrama de flujo de la aplicación de lectura de la tarjeta (Continuación)

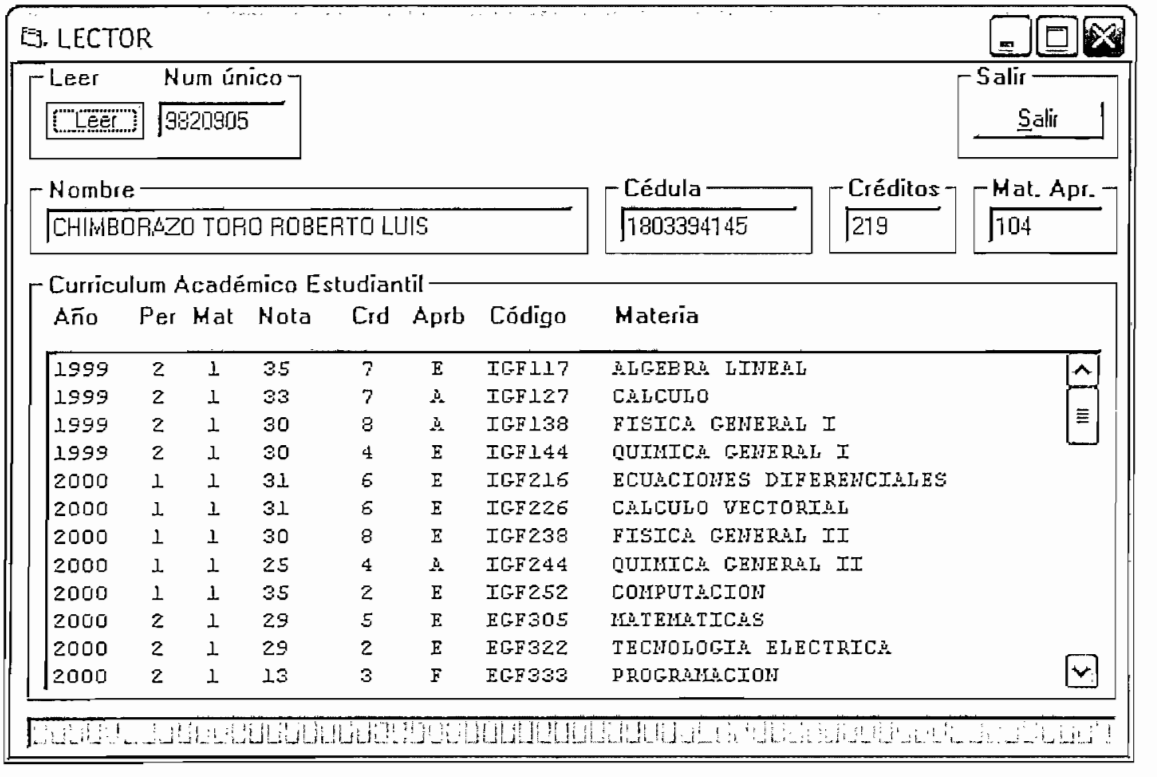

Figura 6.9 Pantalla de lectura exitosa.

6.3.4 PROPIEDADES DE SEGURIDAD

Como se anotó anteriormente como parte de los mecanismos que aseguran que los datos que han sido almacenados en la tarjeta no sean cambiados sin la autorización debida se cuenta con los llamados Bytes de control de errores. Existe un Byte de control de errores para:

- Número único (11 Bytes)
- Número de cédula (12 Bytes)
- Nombre del estudiante (60 Bytes)
- Número de registros, Número de materias aprobadas y Total de créditos acumulados (5 Bytes)
- Cada Registro (5 Bytes)

Esto permite saber si los datos han sido alterados y por tanto mostrar un mensaje que indica esta situación. Además la forma de codificar los datos (materia, nota, créditos, tipo de aprobación, etc.) es tal que los datos no pueden ser intuidos directamente al leer los bytes que conforman los diferentes registros.

Otros mecanismos de seguridad son propios de la tarjeta y son los llamados código de verificación y bits de protección de escritura, como se anota en el Anexo B la tarjeta utilizada cuenta en primer lugar con la característica de protección de escritura, esto significa que cada byte escrito puede ser protegido contra escritura, esto quiere decir que la EEPROM puede ser convertida en ROM. Cave anotar que una vez que se hace uso de esta característica el byte protegido no puede ser cambiando nunca más, pero lógicamente podrá ser leído tantas veces como sea necesario. Al momento esta característica no se está usando en la aplicación dado que para realizar las distintas pruebas no conviene convertir la memoria EEPROM en ROM. Pero en la práctica ésta es una característica que deberá ser utilizada para permitir contar con datos de interés que no deben ser cambiados.

Por otro lado cuando se habla del código de verificación con que cuenta la tarjeta, se habla de un código que debe ser transmitido a la tarjeta para que ésta permita que se puedan escribir o borrar los datos que contiene en su memoria EEPROM.

En primer lugar si se quieren escribir datos en la tarjeta sin tomar en cuenta el código de verificación, dichos datos no son escritos, dado que el modo de funcionamiento de la tarjeta no lo permite. Si por otro lado se quiere adivinar el código de verificación la tarjeta, ésta cuenta con un contador de error que disminuye cada vez que un código de verificación errado es ingresado, haciendo más pequeña la probabilidad de adivinar el código y obtener el acceso con privilegios de escritura a la tarjeta.

Todos estos mecanismos permiten asegurar de alguna forma que la información almacenada puede ser actualizada solamente por personal autorizado o por personas que cuenten con los conocimientos necesarios para hacerlo, dichos conocimientos son:

- Forma de codificación de los datos
- Código de verificación

6.4 DETALLE DE COSTOS

Los costos involucrados en la implementación de esta solución son los siguientes:

Tarjetas SLE4428:	$ 2,30 cada una
Lector y programador:	$ 30

Nota: El costo de las tarjetas disminuye a medida que la cantidad solicitada aumenta, así:

1 a 49	$ 2,30
50 a 99	$ 2,10
100 a 499	$ 1,90
500 a 1000	$ 1,70
1001 a 4999	$ 1,40
más de 4999	$ 1,25

Como se puede apreciar los costos son muy bajos por lo que un estudiante bien podría adquirir una tarjeta de éstas para almacenar su curriculum.

Está claro que existen dos ambientes de trabajo:

El primero es un ambiente en el que el usuario tiene privilegios para escribir la información en las tarjetas. Por lo tanto en este ambiente se debe contar tanto con tarjetas en blanco como con el programador de la tarjeta, las tablas actualizadas que contienen la información referente a todos los alumnos y el software que ofrece la interfaz del usuario.

Mientras que el otro ambiente es aquel en el cual solamente se requiere leer la información de la tarjeta, ya sea que se quiera leer toda la información almacenada en la tarjeta o solo parte de ella se deberá contar con un lector de tarjetas inteligentes y el software que permita realizar la lectura, si se desea una lectura completa además se deberá contar con la tabla que permita decodificar los datos almacenados en la tarjeta. Como se puede apreciar en este ambiente se deberá contar con un lector, cuyo costo es $ 25.

Visto de esta forma se deberán adquirir tantas tarjetas como alumnos estén estudiando en la carrera y el costo de las tarjetas será menor mientras más tarjetas se compren. Por otro lado se requieren tantos lectores-programadores como sitios autorizados para realizar la actualización de información en la tarjeta existan y tantos lectores como sitios donde se requiera acceder a la información almacenada en la tarjeta existan.

CONCLUSIONES Y RECOMENDACIONES

Las tarjetas inteligentes son muy variadas, pero esto no implica dificultad a la hora de usarlas en los diferentes ámbitos de la vida diaria, en otras palabras se han vuelto un elemento tan cotidiano que no se requieren muchos conocimientos para manipular una tarjeta inteligente, esto se debe en parte a los estándares que se relacionan con ellas, los cuales ponen la base sobre la que se sitúa el desempeño total de las tarjetas inteligentes. Los estándares más conocidos son el ISO7816, el cual presenta una base muy sólida en cuanto a las características físicas, propiedades eléctricas y protocolos de transmisión. El estándar EMV que provee un juego de reglas para las tarjetas inteligentes utilizadas en aplicaciones de pago. Además existen muchos otros, los cuales son emitidos por organismos de estandarización mundial, nacional, gubernamental o industrial y que tienen también mucha importancia dependiendo del área de aplicación para la que se está utilizando la tarjeta inteligente.

El camino que han recorrido las tarjetas inteligentes hasta llegar a estos días es muy largo, lo que ha permitido que este elemento sea probado con éxito en muchas aplicaciones y se mantenga aun vigente con mucha demanda. La clave de esto es esencialmente su facilidad de uso, visto desde el punto de vista del usuario final, para quien en muy fácil llevar una tarjeta inteligente consigo e insertarla en un lector (si se trata de una tarjeta con contactos) o simplemente pasar cerca de él (si es una tarjeta sin contactos) para realizar una transacción.

Las características de almacenamiento y procesamiento que presentan las tarjetas inteligentes son muy atractivas, por un lado permiten almacenar mucha más información que una tarjeta de banda magnética, al contar con memorias del tipo EEPROM y hoy en día Flash EEPROM y por el otro realizar procesos con dicha información por si misma, esto por medio de microprocesadores muy versátiles (hardware) junto con sistemas operativos optimizados (software). En

este punto es importante mencionar que en lo que tiene que ver con los microprocesadores la tendencia es llegar a usar microprocesadores de 32 bits, que permitirían contar con tarjetas inteligentes de alto desempeño, mientras que por el lado del sistema operativo se busca optimizarlo al máximo, lo que implica ocupar el menor espacio físico y contar con la mayor funcionalidad.

Cuando se busca intercambiar información con una tarjeta inteligente los protocolos mas utilizados son: un conjunto de **protocolos sincrónicos** muy simples, los cuales principalmente se usan con memory cards, no cuentan con un mecanismo de control de errores y en los cuales no existe separación de capas; el **protocolo** T= 0, es el más usado actualmente, está orientado al byte lo que significa que la unidad más pequeña procesada es un byte, un bit de paridad permite realizar detección de errores; **protocolo** T= 1, es un protocolo orientado al bloque, aquí existe una estricta separación de capas, en este protocolo existen procesos de detección de errores mas elaborados que buscan, luego de que un error es encontrado, restaurar una comunicación libre de errores.

La tarjeta inteligente brinda un ambiente de seguridad para los datos y programas ya que los esfuerzos que un atacante debe hacer para llegar a vulnerar las defensas con que cuentan las tarjetas inteligentes son muchos y cada vez son más. Principalmente la seguridad de una tarjeta inteligente está garantizada por cuatro componentes. El cuerpo de la tarjeta, el hardware del chip, el sistema operativo y la aplicación. Cada uno de estos cuatro componentes cuenta con sus características de seguridad, pero solo si trabajan adecuadamente en conjunto la seguridad física y lógica de la tarjeta inteligente podrá estar garantizada.

La criptografía como un concepto que permite otorgar confidencialidad, integridad y autenticidad ha sido implementada en la tecnología de las tarjetas

inteligentes como una característica de seguridad, para ellos se emplean algoritmos de criptografía como DES y RSA que son los más utilizados. Algunos de estos algoritmos de criptografía por su naturaleza son implementados en hardware (se requiere un módulo), este es el caso de DES, mientras que otros son implementados mediante software (se requiere una rutina).

Hoy en día el tiempo que lleva realizar el diseño e implementación de una solución con tarjetas inteligentes es mucho menor que antes, dado que en la actualidad se cuenta con herramientas muy prácticas para estos fines como el Kit de desarrollo utilizado en este proyecto por ejemplo, además existe una amplia variedad de tarjetas que se ajustan a casi cualquier aplicación en la que se pueda pensar. De igual manera los costos que están asociados con las tarjetas inteligentes cada vez son más bajos esto permite que las aplicaciones de las tarjetas inteligentes puedan llegar a casi cualquier sitio.

Por otra parte la variedad de tarjetas inteligentes es grande, y escoger la tarjeta adecuada para tal o cual aplicación depende de muchos factores como son: lógicamente la experiencia de trabajo con tarjetas inteligentes, los costos que están involucrados, la disponibilidad de las mismas, las características funcionales, etc. Para la realización del proyecto se tomó en cuenta básicamente el costo de las tarjetas inteligentes junto con el costo del lector/programador y la disponibilidad de estos. Paralelamente la tarjeta y lector seleccionados reunieron las características apropiadas para implementar el ejemplo de aplicación.

Cuando se habla de aplicaciones en tarjetas inteligentes, muchas de las veces no solo se habla de la programación de los microprocesadores de las mismas (tarjetas con microprocesador) o de la escritura y lectura de su memoria (memory cards), sino también de otras tecnologías que interactúan con ellas como son: Bases de datos, redes de comunicación, etc. Por esta razón querer implementar una solución con tarjetas inteligentes conlleva tener los conocimientos necesarios de todas las tecnologías que intervendrán en su funcionamiento.

Se recomienda que en un proyecto posterior se realice un estudio enfocado principalmente a las tarjetas inteligentes sin contacto, las cuales presentan la ventaja de no tener que ser insertadas en un lector y por tanto no existen contactos en su superficie que puedan presentar desgaste ni corren el peligro de ser removidas antes de la desactivación de los contactos, estas ventajas las hacen muy atractivas en aplicaciones que se deben realizar en períodos de tiempo muy cortos, como por ejemplo pago de peaje.

Las JavaCards tienen un espacio importante en el mercado de las tarjetas inteligentes, las facilidades de programación en Java y otras ventajas como los mecanismos de seguridad propios de Java hacen de este tipo de tarjetas inteligentes muy atractivas para implementar soluciones con ellas. Me parece importante que se realice un proyecto enfocado a la implementación de una aplicación con JavaCards.

La aplicación aquí descrita puede ser mejorada en muchos sentidos, una primera mejora constituiría la de utilizar otro tipo de tarjeta, talvez una tarjeta inteligente basada en microcontrolador en la cual podrían convivir diversas aplicaciones como por ejemplo, un monedero electrónico que permita adquirir diversos ítems dentro del campus politécnico (comida, libros, tiempo de uso de un computador, copias, etc.), otra que permita que los libros retirados de la biblioteca se registren en la tarjeta, otra más que permita autenticar la identidad de una persona para controlar el acceso a ciertas áreas físicas y virtuales. Esto permitirá contar con un documento que además de identificar a todos quienes forman parte de la E.P.N. otorga otros beneficios extra.

REFERENCIAS BIBLIOGRÁFICAS

[WOLFGANG]
WOLFGANG, Rankl; WOLFGANG, Effing. Smart Card Handbook. Tercera Edición. Editorial John Wiley & Sons, Ltd; Estados Unidos de Norteamérica; 2004.

[GUTHERY]
GUTHERY, Scott; JURGENSEN, Tim. Smart Card Developer's Kit. Editorial Macmillan Computer Publishing; Estados Unidos de Norteamérica; 1998.

[MCKINNEY]
MCKINNEY, Bruce. Programación avanzada con Microsoft Visual Basic 5.0. McGraw Hill, España, 1997.

[FREETUTORIALS]
INTRODUCTION TO SMART CARDS.
www.smartcard.co.uk/downloads/freetutorials.pdf

[NETHANDBOOK]
http://www.nethandbook.com/schist.htm

[CARDWERK]
www.cardwerk.com

[ROOTSHELL]
www.rootshell.be/~**dhar**/downloads/ SmartCards-Introduction.**pdf**

[INGENICO]
http://www.ingenico-ca.com/app/index.cfm?locationid=548&fuseaction=datafile.displayFile

[LIBURUTEGIA]

suse00.su.ehu.es/liburutegia/liburuak/congresos/014/12247255.pdf

[DYDIA]

http://www.didya.com

[ACS]

http://www.acs.com.hk

Anexo A

Líneas de código

En las siguientes líneas se encuentra el código de los programas que permiten escribir la información en la tarjeta, leer la información que ellas contienen, y las líneas de código del módulo utilizado también en ambos programas.

A.1 PROGRAMA DE ESCRITURA (Escritor.frm)

```
'Variables para mostrar registros
    Dim sAno As String
    Dim sPeriodo As String
    Dim sNota As String * 4
    Dim sMateria As String
    Dim sCodMat As String * 6
    Dim sCreditos As String * 2
    Dim sAprobacion As String
    Dim sMatricula As String
'Variables para mostrar direcciones y bytes escritos
    Dim sDireccion As String * 4
    Dim sMSBDir As String * 1
    Dim sLSBDir As String * 2
    Dim sByte1H As String * 2
    Dim sByte2H As String * 2
    Dim sByte3H As String * 2
    Dim sByte4H As String * 2
    Dim sByte5H As String * 2
    Dim sByte6H As String * 2
    Dim sByte1D As String * 3
    Dim sByte2D As String * 3
    Dim sByte3D As String * 3
    Dim sByte4D As String * 3
    Dim sByte5D As String * 3
    Dim sByte6D As String * 3
'Variable para contar materias aprobadas y total de créditos
    Dim MatAprob As Byte
    Dim TotCred As String
'Variables para la base de datos
    Dim WithEvents adoPrimaryRS As Recordset
    Dim mbChangedByCode As Boolean
    Dim mvBookMark As Variant
    Dim mbEditFlag As Boolean
    Dim mbAddNewFlag As Boolean
    Dim mbDataChanged As Boolean
'Variables para comunicación con la tarjeta
    Dim Lector As Integer
    Dim RetCodigo As Long
    Dim Sesion As AC_SESSION
    Dim apdu As AC_APDU
'Variables para Bytes de Registro
    Dim BAno As Byte
    Dim BMatLSB As Byte
    Dim BNotLSB As Byte
    Dim BNotNumMat As Byte
    Dim BSAC As Byte
    Dim RCRC As Byte
'Variables para trabajar con notas y materias
    Dim DNota As Long
    Dim HNota As String
    Dim HNotMSb As String
    Dim HNotLSB As String
```

```
    Dim DNotMSb As Byte
    Dim DNotLSB As Byte
    Dim AuxNotMSb As Byte
    Dim AuxMatMSb As Byte
    Dim BNotNumMater As Byte
    Dim HNotNumMater As String
    Dim HNot As String
    Dim HNumMat As String
    Dim DNot As Byte
    Dim DNumMat As Byte
'Variables para trabajar con Byte SAC (Semestre o período, Aprobación, Créditos)
    Dim HSAC As String
    Dim HSA As String
    Dim HC As String
    Dim DSA As Byte
    Dim DC As Byte
    Dim a As String
    Dim s As Byte
    Dim c As Byte
    Dim Auxs As Byte
    Dim Auxc As Byte
    Dim Auxa As String
    Dim DMatMSb As Byte
    Dim m As Byte
    Dim AuxDNotMSb As Byte
    Dim AuxDMatMSb As Byte
    Dim Auxm As Byte
'Variable auxiliar
    Dim NumReg As Byte
'Variables para la función Bytes de Nombre "ConNombre"
    Dim AuxByte1 As Byte
    Dim AuxByte2 As Byte
'Variables escribir Número único, Nombre y Cédula
    Dim sNunico As String
    Dim sNombre As String
    Dim sCedula As String
    Dim AuxCont As Byte
'==========================================================================================
'Bloque para buscar el curriculum de un alumno luego de haber ingresado su número único
Private Sub Command1_Click()
    Dim db As Connection
    Set db = New Connection
        db.CursorLocation = adUseClient
        db.Open "PROVIDER=MSDASQL;dsn=Escritor;uid=;pwd=;"
    Set adoPrimaryRS = New Recordset
        adoPrimaryRS.Open "SELECT SAEMATER.CODEST,SAEESTUD.NOMEST, SAEESTUD.TOTCRED,
SAEESTUD.CEDULA,SAEMATER.ANOMAT, SAEMATER.CODMAT, SAENOMMA.NOMMAT, SAEMATER.SUMATOM,
SAEMATER.SEMMAT, SAEMATER.NUMMTR, SAEMATER.APRUEBM, SAEMATER.CREDMAT, SAENOMMA.NUMMAT FROM
SAEMATER, SAENOMMA,SAEESTUD WHERE SAEESTUD.CODEST = SAEMATER.CODEST AND SAEMATER.CODMAT =
SAENOMMA.CODMAT AND SAEMATER.CODEST = '" & Text1.Text & "' Order by SAEMATER.ANOMAT,
SAEMATER.SEMMAT,SAEMATER.CODMAT ", db, adOpenStatic, adLockOptimistic
    Dim oText As TextBox
'Enlaza los cuadros de texto con el proveedor de datos
    For Each oText In Me.txtFields
        Set oText.DataSource = adoPrimaryRS
    Next
        mbDataChanged = False
  If txtFields(9).Text = "" Then
        Text8.Text = ""
        List1.Clear
        List2.Clear
        MsgBox "No existe Número único o no hay materias registradas!!!", vbOKOnly,"ERROR"
        Exit Sub
  Else
        Frame2.Visible = True
        Command2.Visible = True
        Command1.Enabled = False
        List1.Clear
        List2.Clear
  End If
'Mostrar en el primer list los registros de materias tomadas por el estudiante, luego de
'haber verificado que su número único existe y tiene materias registradas en la base de
'datos

    MatAprob = 0
```

```
  If adoPrimaryRS.RecordCount < 141 Then
        For NumReg = 1 To adoPrimaryRS.RecordCount
            'Contar materias aprobadas
            If txtFields(8).Text = "A" Or txtFields(8).Text = "E" Then
                MatAprob = MatAprob + 1
            End If
            sAno = CStr(txtFields(0))
            sCodMat = CStr(txtFields(1))
            sMateria = CStr(txtFields(2))
            sNota = CStr(txtFields(3))
            sPeriodo = CStr(txtFields(4))
            sCreditos = CStr(txtFields(6))
            sMatricula = CStr(txtFields(7))
            sAprobacion = CStr(txtFields(8))
            List1.AddItem (sAno + "    " + sPeriodo + "    " + sMatricula + "    " + sNota + "
" + sCreditos + "    " + sAprobacion + "    " + sCodMat + "    " + sMateria)
            'Pase al siguiente registro
            Call Siguiente
        Next NumReg

        Text8.Text = MatAprob
        Call Primero
    Else
        List1.AddItem ("¡¡¡ El número de registros es mayor que 140 !!!")
        Exit Sub
    End If
End Sub
'=========================================================================================
'Bloque para escribir en la tarjeta toda la información correspondiente al curriculum
Private Sub Command2_Click()

    'Abrir el puerto
        Lector = AC_Open(ACR20_115200, 0)    'Velocidad 115200 kbps, puerto: COM 1
        If Lector < 0 Then
            MsgBox "Error al abrir puerto COM 1"
        Exit Sub
        End If

    'Seleccionar el tipo de tarjeta SLE 4428
        Sesion.CardType = AC_AM8KS                    'Tipo SLE 4428
        RetCodigo = AC_StartSession(Lector, Sesion)
        If (RetCodigo < 0) Then
            MsgBox "Error al iniciar sesión"
        Exit Sub
        End If

    'Código de Verificación para poder escribir en la tarjeta
        apdu.CLA = &H0            'Instrucción de clase
        apdu.INS = ACI_Verify     'Comando de Verificación
        apdu.P1 = &H0             'No importa
        apdu.P2 = &H0             'No importa
        apdu.Lc = &H2             'Código de transporte 2 byte (# de bytes de la llave)
        apdu.Le = &H3             '&H3 es el valor en este campo para esta tarjeta
        apdu.DataIn(1) = &HFF     'Código de trasporte por defecto 0xFF
        apdu.DataIn(2) = &HFF     'Código de trasporte por defecto 0xFF

       'Se transmite el comando a la tarjeta insertada
        RetCodigo = AC_ExchangeAPDU(Lector, apdu)
        If RetCodigo < 0 Then             'Si Retcodigo es negativo se ha producido un error
            MsgBox "Error en código de verificación"                    'Mensaje de error
        Exit Sub
        End If
'=========================================================================================
    'Bloque para escribir Número Único
        sNunico = Text1.Text
        apdu.CLA = &H0            'Instrucción de clase
        apdu.INS = ACI_Write      'Código de instrucción Comando de escritura
        apdu.P1 = 0               'MSB de la dirección a ser escrita
        apdu.P2 = 27              'LSB de la dirección a ser escrita
        apdu.Lc = 12              'Número de Bytes a ser escritos en la tarjeta
        apdu.Le = &H0             'No se espera ninguna respuesta de la tarjeta
        apdu.DataIn(12) = 0
```

```
    'Escribir el número único como una cadena de caracteres
        For AuxCont = 1 To 11
            If AuxCont <= Len(sNunico) Then
                apdu.DataIn(AuxCont) = AscB(Mid(sNunico, AuxCont, 1))        'Letra x letra
            Else
                apdu.DataIn(AuxCont) = AscB(" ")
            End If
            apdu.DataIn(12) = apdu.DataIn(12) Xor apdu.DataIn(AuxCont)  'CRC de Núm. unico
        Next AuxCont

    'Se transmite el comando a la tarjeta insertada
        RetCodigo = AC_ExchangeAPDU(Lector, apdu)
        If RetCodigo < 0 Then            'Si Retcodigo es negativo se ha producido un error
            MsgBox "Fallo al escribir Número único"                      'Mensaje de error
        Exit Sub
        End If

'=============================================================================================
    'Bloque para escribir Cédula
        sCedula = txtFields(10).Text
        apdu.CLA = &H0          'Instrucción de clase
        apdu.INS = ACI_Write    'Código de instrucción Comando de escritura
        apdu.P1 = 0             'MSB de la dirección a ser escrita
        apdu.P2 = 39            'LSB de la dirección a ser escrita
        apdu.Lc = 13            'Número de Bytes a ser escritos en la tarjeta
        apdu.Le = &H0           'No se espera ninguna respuesta de la tarjeta
        apdu.DataIn(13) = 0

        If Left(sCedula, 1) = " " Then
            For AuxCont = 1 To 12
                'Escribir xxxx cuando no existe # de cédula
                apdu.DataIn(AuxCont) = AscB("X")
                apdu.DataIn(13) = apdu.DataIn(13) Xor apdu.DataIn(AuxCont) 'CRC
            Next AuxCont
        Else
            For AuxCont = 1 To 12
                If AuxCont <= Len(sCedula) Then
                    'Bye/Byte o Letra/Letra
                    apdu.DataIn(AuxCont) = AscB(Mid(sCedula, AuxCont, 1))
                Else
                    apdu.DataIn(AuxCont) = AscB(" ")
                End If
                apdu.DataIn(13) = apdu.DataIn(13) Xor apdu.DataIn(AuxCont)  'CRC
            Next AuxCont
        End If

    'Se transmite el comando a la tarjeta insertada
        RetCodigo = AC_ExchangeAPDU(Lector, apdu)
        If RetCodigo < 0 Then            'Si Retcodigo es negativo se ha producido un error
            MsgBox "Error al escribir Número de Cédula"                  'Mensaje de error
        Exit Sub
        End If

'=============================================================================================
    'Bloque para escribir Nombre
        sNombre = txtFields(9).Text
        apdu.CLA = &H0          'Instrucción de clase
        apdu.INS = ACI_Write    'Código de instrucción Comando de escritura
        apdu.P1 = &H0           'MSB de la dirección a ser escrita
        apdu.P2 = 52            'LSB de la dirección a ser escrita
        apdu.Lc = 61            'Número de Bytes a ser escritos en la tarjeta
        apdu.Le = &H0           'No se espera ninguna respuesta de la tarjeta

    'Inicializar CRC
        apdu.DataIn(61) = 0

    'Copiar cada caracter en el respectivo Apdu.DataIn
        For AuxCont = 1 To 60
            If AuxCont <= Len(sNombre) Then
                apdu.DataIn(AuxCont) = AscB(Mid(sNombre, AuxCont, 1))
            Else
                apdu.DataIn(AuxCont) = AscB(" ")
            End If
            apdu.DataIn(61) = apdu.DataIn(61) Xor apdu.DataIn(AuxCont)
        Next AuxCont
```

```
    'Se transmite el comando a la tarjeta insertada
        RetCodigo = AC_ExchangeAPDU(Lector, apdu)
        If RetCodigo < 0 Then            'Si Retcodigo es negativo se ha producido un error
            MsgBox "Error al escribir Nombre"                            'Mensaje de error
            Exit Sub
        End If
'==========================================================================================
    'Bloque para escribir registros
    If adoPrimaryRS.RecordCount < 141 Then 'Controlar no más de 140 registros
        PBar.Max = adoPrimaryRS.RecordCount
        PBar.Value = 0
        For NumReg = 1 To adoPrimaryRS.RecordCount
            BAno = Val(txtFields(0).Text) - 1900    'Byte de Año
            WriteNota (txtFields(3).Text * 10)      'Función para escribir Nota
            BNotLSB = DNotLSB                       'Byte Menos significante de Nota
            AuxNotMSb = DNotMSb                     'Auxiliar para Nota
            WriteNota (txtFields(5).Text)           'Función para escribir Materia
            BMatLSB = DNotLSB                       'Byte Menos significante de Materia
            AuxMatMSb = DNotMSb                     'Auxiliar para Materia
           'Función para construir Byte con información de período, aprobación y créditos
            BSAC = WriteSAC(txtFields(4).Text, txtFields(8).Text, txtFields(6).Text)
           'Función para construir Byte con información de Nota, Materia y número de
           'materia
            BNotNumMat = WriteNotNumMater(AuxNotMSb, AuxMatMSb, txtFields(7).Text)
           'Byte con código de verificación de datos
            BCRC = BAno Xor BNotNumMat Xor BMatLSB Xor BNotLSB Xor BSAC
        'Almaceno en DirReg el valor de la dirección a ser escrita en base al # de registro
            DirReg = 6 * (NumReg - 1) + 174
        'Recuperar la dirección en forma de 2 bytes con ayuda de la función CodNombre
            CodNombre (DirReg)
        'Bloque para escribir un registro y CRC
            apdu.CLA = &H0              'Instrucción de clase
            apdu.INS = ACI_Write        'Código de instrucción Comando de escritura
            apdu.P1 = AuxByte1          'MSB de la dirección a ser escrita
            apdu.P2 = AuxByte2          'LSB de la dirección a ser escrita
            apdu.Lc = &H6               'Número de Bytes a ser escritos en la tarjeta
            apdu.Le = &H0               'No se espera ninguna respuesta de la tarjeta
            apdu.DataIn(1) = BAno
            apdu.DataIn(2) = BNotNumMat
            apdu.DataIn(3) = BMatLSB
            apdu.DataIn(4) = BNotLSB
            apdu.DataIn(5) = BSAC
            apdu.DataIn(6) = BCRC

        'Se transmite el comando a la tarjeta insertada
            RetCodigo = AC_ExchangeAPDU(Lector, apdu)
            If RetCodigo < 0 Then        'Si Retcodigo es negativo se ha producido un error
                MsgBox "Fallo al escribir Registro"                      'Mensaje de error
                Exit Sub
            End If

        'Escribir en list2 los registros escritos en forma de números hexadecimales
        sDireccion = CStr(DirReg)
        sMSBDir = CStr(Hex(apdu.P1))
        sLSBDir = CStr(Hex(apdu.P2))
        sByte1H = CStr(Hex(apdu.DataIn(1)))
        sByte2H = CStr(Hex(apdu.DataIn(2)))
        sByte3H = CStr(Hex(apdu.DataIn(3)))
        sByte4H = CStr(Hex(apdu.DataIn(4)))
        sByte5H = CStr(Hex(apdu.DataIn(5)))
        sByte6H = CStr(Hex(apdu.DataIn(6)))
        sByte1D = CStr(apdu.DataIn(1))
        sByte2D = CStr(apdu.DataIn(2))
        sByte3D = CStr(apdu.DataIn(3))
        sByte4D = CStr(apdu.DataIn(4))
        sByte5D = CStr(apdu.DataIn(5))
        sByte6D = CStr(apdu.DataIn(6))
        List2.AddItem (sMSBDir + sLSBDir + "= " + sDireccion + " <=(" + sByte1H + " =" +
sByte1D + ") (" + sByte2H + " =" + sByte2D + ") (" + sByte3H + " =" + sByte3D + ") (" +
sByte4H + " =" + sByte4D + ") (" + sByte5H + " =" + sByte5D + ") =(" + sByte6H + " =" +
sByte6D + ")")

        'Pase al siguiente registro
        Call Siguiente
        PBar.Value = PBar.Value + 1
```

```
        Next NumReg
    Else
        List2.AddItem ("¡¡¡ El numero de registros es mayor que 140 !!!")
        Exit Sub
    End If

'=========================================================================================
    'Bloque para escribir # de registros, # de materias aprobadas, Total de créditos y
    'Verificación de escritura
        TotCred = CStr(txtFields(11).Text)
        If Len(TotCred) = 1 Then
            TotCred = "00" + TotCred
        ElseIf Len(TotCred) = 2 Then
            TotCred = "0" + TotCred
        End If
        Frame7.Visible = True
        Text8.Visible = True
        apdu.CLA = &H0                  'Instrucción de clase
        apdu.INS = ACI_Write            'Código de instrucción Comando de escritura
        apdu.P1 = 0                     'MSB de la dirección a ser escrita
        apdu.P2 = 113                   'LSB de la dirección a ser escrita
        apdu.Lc = 8                     'Número de Bytes a ser escritos en la tarjeta
        apdu.Le = &H0                   'No se espera ninguna respuesta de la tarjeta
        apdu.DataIn(1) = adoPrimaryRS.RecordCount
        apdu.DataIn(2) = MatAprob
        apdu.DataIn(3) = AscB(Mid(TotCred, 1, 1))
        apdu.DataIn(4) = AscB(Mid(TotCred, 2, 1))
        apdu.DataIn(5) = AscB(Mid(TotCred, 3, 1))
        apdu.DataIn(6) = AscB("S")
        apdu.DataIn(7) = AscB("i")
        apdu.DataIn(8) = apdu.DataIn(1) Xor apdu.DataIn(2) Xor apdu.DataIn(3) Xor
apdu.DataIn(4) Xor apdu.DataIn(5) Xor apdu.DataIn(6) Xor apdu.DataIn(7)
    'Se transmite el comando a la tarjeta insertada
        RetCodigo = AC_ExchangeAPDU(Lector, apdu)
        If RetCodigo < 0 Then            'Si Retcodigo es negativo se ha producido un error
            MsgBox "Fallo al escribir # de creditos"                    'Mensaje de error
        Exit Sub
        End If
    'Se muestra número de registros, número de materias aprobadas, total de créditos
        MsgBox "Escritura Satisfactoria !!!", vbOKOnly, "OK !!!"
        Call Form_Load
End Sub
'=========================================================================================
'Bloque para cerrar el programa
Private Sub Command3_Click()
    Unload Me
End Sub
'=========================================================================================
'Bloque para desactivar botón grabar al iniciar el programa
Private Sub Form_Load()
    Command1.Enabled = True
    Frame2.Visible = False
    Command2.Visible = False
End Sub
'=========================================================================================
'Bloque para cerrar programa
Private Sub Form_Unload(Cancel As Integer)
    RetCodigo = AC_EndSession(Lector)
    RetCodigo = AC_Close(Lector)
    Screen.MousePointer = vbDefault
End Sub
'=========================================================================================
'Bloque para avanzar al siguiente registro del curriculum
Private Sub Siguiente()
    On Error GoTo GoNextError

    If Not adoPrimaryRS.EOF Then
        adoPrimaryRS.MoveNext
        If adoPrimaryRS.EOF And adoPrimaryRS.RecordCount > 0 Then
            adoPrimaryRS.MoveLast
        End If
    End If
    'Muestra el registro actual
    mbDataChanged = False
```

```
    Exit Sub
GoNextError:
    MsgBox Err.Description
End Sub
'==========================================================================================
'Función WriteNota, permite el ingreso de un número en forma decimal entre 0 y 4095, y lo
'separa en dos bytes, en la variable DNotMSb está almacenado el Byte más significante en
'forma decimal y en la variable DNotLSB el Byte menos significante tambien en forma
'decimal.
Sub WriteNota(DNota As Long)
    HNota = CStr(Hex(DNota))
    If Len(HNota) = 1 Then
        HNota = "00" + HNota
    ElseIf Len(HNota) = 2 Then
        HNota = "0" + HNota
    End If
    HNotMSb = Left(HNota, 1)
    HNotLSB = Right(HNota, 2)
    DNotMSb = Val("&h" & HNotMSb)
    DNotLSB = Val("&h" & HNotLSB)
End Sub

'==========================================================================================
'La Function WriteSAC construye un byte a partir de tres variables que son las siguientes:
's, es un entero que puede variar entre 1 y 4
'a, es una letra que puede ser: E, A, F o X
'c, es un entero que puede variar entre puede ser 0,1,2,3,4,5,6,7,8 o 20
Function WriteSAC(s As Byte, a As String, c As Byte) As Byte
    Auxs = s
    Auxc = c
    Auxa = a
    'Lazo para comparar SAC ingresado con todas las posibilidades
    For WriteSAC = 0 To 255
        'Convertir el decimal en Hex y rellenar
        HSAC = CStr(Hex(WriteSAC))
        If Len(HSAC) = 1 Then
            HSAC = "0" + HSAC
        End If
        'Separar en dos Nibles
        HSA = Left(HSAC, 1)
        HC = Right(HSAC, 1)
        DSA = Val("&h" & HSA)
        DC = Val("&h" & HC)

        If DSA > 11 And DSA < 16 Then
            a = "X"
            s = DSA - 11
        ElseIf DSA > 7 And DSA < 12 Then
            a = "F"
            s = DSA - 7
        ElseIf DSA > 3 And DSA < 8 Then
            a = "A"
            s = DSA - 3
        Else
            a = "E"
            s = DSA + 1
        End If

        If DC = 15 Then
            c = 20
        Else
            c = DC
        End If
        If Auxs = s And Auxc = c And Auxa = a Then
            Exit Function
        End If
    Next WriteSAC
End Function
'==========================================================================================
'La función WriteNotNumMater construye un byte a partir de tres variables que son las
'siguientes:
'La primera es un entero que varia entre 0 y 1
'la segunda es un entero que varia entre 0 y 3
'y la tercera es un entero que varia entre 1 y 4
Function WriteNotNumMater(DNotMSb As Byte, DMatMSb As Byte, m As Byte) As Byte
```

```
    AuxDNotMSb = DNotMSb
    AuxDMatMSb = DMatMSb
    Auxm = m
    For WriteNotNumMater = 0 To 255
        'Convierto Decimal en HEX y relleno
        HNotNumMater = CStr(Hex(WriteNotNumMater))
        If Len(HNotNumMater) = 1 Then
            HNotNumMater = "0" + HNotNumMater
        End If
        HNot = Left(HNotNumMater, 1)
        HNumMat = Right(HNotNumMater, 1)
        DNot = Val("&h" & HNot)
        DNumMat = Val("&h" & HNumMat)

        If DNot > 7 Then
            DNotMSb = 1
        Else
            DNotMSb = 0
        End If

        If DNumMat > 11 And DNumMat < 16 Then
            DMatMSb = 3
            m = DNumMat - 11
        ElseIf DNumMat > 7 And DNumMat < 12 Then
            DMatMSb = 2
            m = DNumMat - 7
        ElseIf DNumMat > 3 And DNumMat < 8 Then
            DMatMSb = 1
            m = DNumMat - 3
        Else
            DMatMSb = 0
            m = DNumMat + 1
        End If
        If AuxDNotMSb = DNotMSb And AuxDMatMSb = DMatMSb And Auxm = m Then
            Exit Function
        End If
    Next WriteNotNumMater
End Function
'==========================================================================================
'La función CodNombre convierte un decimal en dos decimales, en AuxByte1 se almacena el
'byte mas significante del número ingresado y en AuxByte2 el menos significante
Sub CodNombre(NombreDec As Long)
    Dim AuxByte As String
    Dim a(2) As String
    Dim i As Integer
    AuxByte = CStr(Hex(NombreDec))
    If Len(AuxByte) = 3 Then
        AuxByte = "0" + AuxByte
    ElseIf Len(AuxByte) = 2 Then
        AuxByte = "00" + AuxByte
    ElseIf Len(AuxByte) = 1 Then
        AuxByte = "000" + AuxByte
    End If

    For i = 1 To 2                              'Almacena los caracteres uno a uno en
        a(i - 1) = Mid(AuxByte, i * 2 - 1, 2)   'el vector si la cadena es "123456"
    Next i                                      'A(0)=1,A(1)=2,A(2)=3 ...

    AuxByte1 = Val("&h" & a(0))    'Almacena el valor del MSB en decimal
    AuxByte2 = Val("&h" & a(1))    'Almacena el valor del LSB en decimal
End Sub
'==========================================================================================
'Función para ir al primer registro del curriculum
Private Sub Primero()
  On Error GoTo GoFirstError
    adoPrimaryRS.MoveFirst
    mbDataChanged = False
  Exit Sub
GoFirstError:
    MsgBox Err.Description
End Sub
```

A.2 PROGRAMA DE LECTURA (Lector.frm)

```
'Variables para mostrar registros
    Dim sAno As String
    Dim sPeriodo As String
    Dim sNota As String * 4
    Dim sMateria As String
    Dim sCodMat As String * 6
    Dim sCreditos As String * 2
    Dim sAprobacion As String
    Dim sMatricula As String
'Variables para comunicación con la tarjeta
    Dim Lector As Integer
    Dim RetCodigo As Long
    Dim Sesion As AC_SESSION
    Dim apdu As AC_APDU
    Dim aux As Byte
'Variables para la funcion Bytes de Nombre ConNombre
    Dim AuxByte1 As Byte
    Dim AuxByte2 As Byte
'Variables para la base de datos
    Dim WithEvents adoPrimaryRS As Recordset
    Dim mbChangedByCode As Boolean
    Dim mvBookMark As Variant
    Dim mbEditFlag As Boolean
    Dim mbAddNewFlag As Boolean
    Dim mbDataChanged As Boolean
'Variables nuevas
    Dim BNotNumMater As Byte
    Dim HNotNumMater As String
    Dim HNot As String
    Dim HNumMat As String
    Dim DNot As Byte
    Dim DNumMat As Byte
    Dim DNotMSb As Byte
    Dim DMatMSb As Byte
    Dim m As Byte
    Dim AuxDNotMSb As Byte
    Dim AuxDMatMSb As Byte
    Dim Auxm As Byte
    Dim DNota As Long
    Dim HNota As String
    Dim HNotMSb As String
    Dim HNotLSB As String
    Dim DNotLSB As Byte
    Dim Nota As Long
    Dim Materia As Long
'Variables para trabajar con Byte SAC (Semestre, Aprobación, Créditos)
    Dim BSAC As Byte
    Dim HSAC As String
    Dim HSA As String
    Dim HC As String
    Dim DSA As Byte
    Dim DC As Byte
    Dim a As String
    Dim s As Byte
    Dim c As Byte
    Dim Auxs As Byte
    Dim Auxc As Byte
    Dim Auxa As String
'Variable para verificación
    Dim Verificacion As Integer
'==========================================================================================
'Bloque para leer la información de la tarjeta
Private Sub Command1_Click()
    'Abrir el puerto
        Lector = AC_Open(ACR20_115200, 0)            'Velocidad 115200 kbps, puerto: COM 1
        If Lector < 0 Then
            MsgBox "Error al abrir Puerto"
        Exit Sub
        End If
```

```
'Seleccionar el tipo de tarjeta SLE 4428
    Sesion.CardType = AC_AM8KS                                          'Tipo SLE 4428
    RetCodigo = AC_StartSession(Lector, Sesion)
    If (RetCodigo < 0) Then
        MsgBox "Fallo al iniciar Sesión"
    Exit Sub
    End If
'Bloque para Verificar escritura
    apdu.CLA = &H0              'Instrucción de clase
    apdu.INS = ACI_Read         'Código de instrucción Comando de Lectura
    apdu.P1 = &H0               'MSB del inicio de la dirección a ser Leída
    apdu.P2 = 118               'LSB del inicio de la dirección a ser Leída
    apdu.Lc = &H0               'No es necesario
    apdu.Le = &H2               'Lea los primeros 2 bytes!
    RetCodigo = AC_ExchangeAPDU(Lector, apdu)                'Ejecución del comando leer
    If RetCodigo < 0 Then             'Si Retcodigo es negativo se ha producido un error
        MsgBox "Error al leer Cédula"                                   'Mensaje de error
    Exit Sub
    End If
'Almacenar lo leído en sNunico
    sNunico = ""
    For AuxCont = 1 To 2
        sNunico = sNunico + Chr(apdu.DataOut(AuxCont))
    Next AuxCont
'Comprobar si la tarjeta ya ha sido escrita o no
If sNunico = "Si" Then
    'Bloque para leer Número Único
        apdu.CLA = &H0              'Instrucción de clase
        apdu.INS = ACI_Read         'Código de instrucción Comando de Lectura
        apdu.P1 = 0                 'MSB del inicio de la dirección a ser Leída
        apdu.P2 = 27                'LSB del inicio de la dirección a ser Leída
        apdu.Lc = &H0               'No es necesario
        apdu.Le = 12                'Lea los primeros 12 bytes!
        RetCodigo = AC_ExchangeAPDU(Lector, apdu)            'Ejecución del comando leer
        If RetCodigo < 0 Then         'Si Retcodigo es negativo se ha producido un error
            MsgBox "Error al leer Número único"                         'Mensaje de error
        Exit Sub
        End If
    'Almacenar lo leído en la variable sNunico
        sNunico = ""
        Verificacion = 0
        For AuxCont = 1 To 11
            sNunico = sNunico + Chr(apdu.DataOut(AuxCont))
            Verificacion = apdu.DataOut(AuxCont) Xor Verificacion
        Next AuxCont
     'Bloque de verificación
        If Verificacion = apdu.DataOut(12) Then
            Text1.Text = sNunico
        Else
            MsgBox "Número único adulterado"                            'Mensaje de error
            Exit Sub
        End If
    'Bloque para leer Cédula
        apdu.CLA = &H0              'Instrucción de clase
        apdu.INS = ACI_Read         'Código de instrucción Comando de Lectura
        apdu.P1 = &H0               'MSB del inicio de la dirección a ser Leída
        apdu.P2 = 39                'LSB del inicio de la dirección a ser Leída
        apdu.Lc = &H0               'No es necesario
        apdu.Le = 13                'Lea los primeros 13 bytes!
        RetCodigo = AC_ExchangeAPDU(Lector, apdu)            'Ejecución del comando leer
        If RetCodigo < 0 Then         'Si Retcodigo es negativo se ha producido un error
            MsgBox "Error al leer Cédula"                               'Mensaje de error
        Exit Sub
        End If

    'Almacenar lo leído en la variable sNunico
        sNunico = ""
        Verificacion = 0
        For AuxCont = 1 To 12
            sNunico = sNunico + Chr(apdu.DataOut(AuxCont))
            Verificacion = apdu.DataOut(AuxCont) Xor Verificacion
        Next AuxCont
```

```
        'Bloque de verificacion
            If Verificacion = apdu.DataOut(13) Then
                Text3.Text = sNunico
            Else
                MsgBox "Número de Cédula adulterado"                    'Mensaje de error
                Exit Sub
            End If
        'Bloque para leer Nombre
            apdu.CLA = &H0              'Instrucción de clase
            apdu.INS = ACI_Read         'Código de instrucción Comando de Lectura
            apdu.P1 = 0                 'MSB del inicio de la dirección a ser Leída
            apdu.P2 = 52                'LSB del inicio de la dirección a ser Leída
            apdu.Lc = &H0               'No es necesario
            apdu.Le = 61                'Lea los primeros 61 bytes!
            RetCodigo = AC_ExchangeAPDU(Lector, apdu)           'Ejecución del comando leer
            If RetCodigo < 0 Then         'Si Retcodigo es negativo se a producido un error
                MsgBox "Error al leer Nombre"                           'Mensaje de error
            Exit Sub
            End If
        'Almacenar lo leído en la variable sNunico
            sNunico = ""
            Verificacion = 0
            For AuxCont = 1 To 60
                sNunico = sNunico + Chr(apdu.DataOut(AuxCont))
                Verificacion = apdu.DataOut(AuxCont) Xor Verificacion
            Next AuxCont
        'Bloque de verificación
            If Verificacion = apdu.DataOut(61) Then
                Text2.Text = sNunico
            Else
                MsgBox "Nombre del alumno adulterado"                   'Mensaje de error
                Exit Sub
            End If
        'Bloque para Créditos
            apdu.CLA = &H0              'Instrucción de clase
            apdu.INS = ACI_Read         'Código de instrucción Comando de Lectura
            apdu.P1 = 0                 'MSB del inicio de la dirección a ser Leída
            apdu.P2 = 113               'LSB del inicio de la dirección a ser Leída
            apdu.Lc = &H0               'No es necesario
            apdu.Le = 8                 'Lea los primeros 8 bytes!
            RetCodigo = AC_ExchangeAPDU(Lector, apdu)           'Ejecución del comando leer
            If RetCodigo < 0 Then        'Si Retcodigo es negativo se ha producido un error
                MsgBox "Error al leer créditos"                         'Mensaje de error
            Exit Sub
            End If
        'Bloque de verificación
            Verificacion = 0
            For AuxCont = 1 To 7
                Verificacion = apdu.DataOut(AuxCont) Xor Verificacion
            Next AuxCont
            If Verificacion = apdu.DataOut(8) Then
                NumReg1 = apdu.DataOut(1)
                mataprob = apdu.DataOut(2)
                totcred = Val(Chr(apdu.DataOut(3)) + Chr(apdu.DataOut(4)) +
Chr(apdu.DataOut(5)))
                Text4.Text = totcred
                Text5.Text = mataprob
            Else
                MsgBox "Número de Créditos adulterado"                  'Mensaje de error
                Exit Sub
            End If


        'Bloque para leer registros
            PBar.Max = NumReg1
            PBar.Value = 0
            For NumReg = 1 To NumReg1
                'Almaceno en DirReg el valor de la dirección a ser escrita en base al
                '# de registro
                    DirReg = 6 * (NumReg - 1) + 174

                'Recuperar la dirección en forma de 2 bytes
                    CodNombre (DirReg)
```

```
                'Bloque para leer Registro
                    apdu.CLA = &H0              'Instrucción de clase
                    apdu.INS = ACI_Read         'Código de instrucción Comando de Lectura
                    apdu.P1 = AuxByte1          'MSB del inicio de la dirección a ser Leída
                    apdu.P2 = AuxByte2          'LSB del inicio de la dirección a ser Leída
                    apdu.Lc = &H0               'No es necesario
                    apdu.Le = &H6               'Lea los primeros 6 bytes!
                    RetCodigo = AC_ExchangeAPDU(Lector, apdu)   'Ejecucion del comando leer
                    If RetCodigo <0 Then 'Si Retcodigo es negativo se ha producido un error
                        MsgBox "Error al leer Registro"                  'Mensaje de error
                    Exit Sub
                    End If

                'Crear Byte de verificación
                    Verificacion = 0
                    For AuxCont = 1 To 5
                        Verificacion = apdu.DataOut(AuxCont) Xor Verificacion
                    Next AuxCont

                'Bloque de verificación
                    If Verificacion = apdu.DataOut(6) Then
                        Call ReadNotNumMater(apdu.DataOut(2))
                        Call ReadSAC(apdu.DataOut(5))
                        Nota = ReadNota(DNotMSb, apdu.DataOut(4))
                        Materia = ReadNota(DMatMSb, apdu.DataOut(3))
                        Call Command2_Click
                        sAno = CStr(apdu.DataOut(1) + 1900)
                        sCodMat = txtFields(6).Text
                        sMateria = txtFields(5).Text
                        sNota = CStr(Nota / 10)
                        sPeriodo = CStr(s)
                        sCreditos = CStr(c)
                        sMatricula = CStr(m)
                        sAprobacion = a
                        List1.AddItem (sAno + "   " + sPeriodo + "   " + sMatricula + "   "
+ sNota + "   " + sCreditos + "   " + sAprobacion + "   " + sCodMat + "   " + sMateria)
                        PBar.Value = PBar.Value + 1
                    Else
                        MsgBox "Registro " + CStr(NumReg) + "adulterado"
                        Exit Sub
                    End If
                Next NumReg
        Else
            MsgBox "Tarjeta no Escrita"
        End If
End Sub

'==========================================================================================
'Función para Recuperar el nombre de una materia de la tabla NUMMAT
Private Sub Command2_Click()
  Dim db As Connection
  Set db = New Connection
  db.CursorLocation = adUseClient
  db.Open "PROVIDER=MSDASQL;dsn=Lector;uid=;pwd=;"

  Set adoPrimaryRS = New Recordset
  adoPrimaryRS.Open "select CODMAT,NOMMAT,NUMMAT,CREDITOSE,CATCRE from SAENOMMA where
NUMMAT like '" & CStr(Materia) & "' ", db, adOpenStatic, adLockOptimistic
  Dim oText As TextBox

  'Enlaza los cuadros de texto con el proveedor de datos
  For Each oText In Me.txtFields
    Set oText.DataSource = adoPrimaryRS
  Next

  mbDataChanged = False
  db.Close
End Sub
'==========================================================================================
'Cerrar el programa
Private Sub Command3_Click()
    Unload Me
End Sub

'==========================================================================================
```

```
'Cerrar el programa
Private Sub Form_Unload(Cancel As Integer)
    RetCodigo = AC_EndSession(Lector)
    RetCodigo = AC_Close(Lector)
End Sub

'==========================================================================================
'La funcion CodNombre convierte un decimal en dos decimales, en AuxByte1 se almacena el
'byte mas significante del número ingresado y en AuxByte2 el menos significante
Sub CodNombre(NombreDec As Long)
    Dim AuxByte As String
    Dim a(2) As String
    Dim i As Integer
    AuxByte = CStr(Hex(NombreDec))
    If Len(AuxByte) = 3 Then
        AuxByte = "0" + AuxByte
    ElseIf Len(AuxByte) = 2 Then
        AuxByte = "00" + AuxByte
    ElseIf Len(AuxByte) = 1 Then
        AuxByte = "000" + AuxByte
    Else
    End If

    For i = 1 To 2                                  'Almacena los caracteres uno a uno en
        a(i - 1) = Mid(AuxByte, i * 2 - 1, 2)        'el vector si la cadena es "123456"
    Next i                                                    'A(0)=1,A(1)=2,A(2)=3 ...
    AuxByte1 = Val("&h" & a(0))                     'Almacena el valor del MSB en decimal
    AuxByte2 = Val("&h" & a(1))                     'Almacena el valor del LSB en decimal
End Sub

'==========================================================================================
'Función que permite recuperar el bit más significante de Nota (DNotMSb), los dos bits mas
'significantes para decodificar materia (DMatMSb), y el número de matrícula (m)
Sub ReadNotNumMater(BNotNumMater As Byte)
HNotNumMater = CStr(Hex(BNotNumMater))
If Len(HNotNumMater) = 1 Then
    HNotNumMater = "0" + HNotNumMater
End If
HNot = Left(HNotNumMater, 1)
HNumMat = Right(HNotNumMater, 1)
DNot = Val("&h" & HNot)
DNumMat = Val("&h" & HNumMat)

If DNot > 7 Then
    DNotMSb = 1
Else
    DNotMSb = 0
End If

If DNumMat > 11 And DNumMat < 16 Then
    DMatMSb = 3
    m = DNumMat - 11
ElseIf DNumMat > 7 And DNumMat < 12 Then
    DMatMSb = 2
    m = DNumMat - 7
ElseIf DNumMat > 3 And DNumMat < 8 Then
    DMatMSb = 1
    m = DNumMat - 3
Else
    DMatMSb = 0
    m = DNumMat + 1
End If
End Sub

'==========================================================================================
'Esta función permite recuperar la nota almacenada en la tarjeta para cada registro
Function ReadNota(DNotMSb As Byte, DNotLSB As Byte) As Long
    HNotLSB = CStr(Hex(DNotLSB))
    If Len(HNotLSB) = 1 Then
        HNotLSB = "0" + HNotLSB
    End If
    HNota = CStr(Hex(DNotMSb)) + HNotLSB
    ReadNota = Val("&h" & HNota)
End Function
```

```
'==========================================================================================
'Esta función permite recuperar leer de la tarjeta el periodo, el tipo de aprobación y el
'número de créditos relacionados con la materia
Sub ReadSAC(BSAC As Byte)
    HSAC = CStr(Hex(BSAC))
    If Len(HSAC) = 1 Then
        HSAC = "0" + HSAC
    End If
    HSA = Left(HSAC, 1)
    HC = Right(HSAC, 1)
    DSA = Val("&h" & HSA)
    DC = Val("&h" & HC)
    If DSA > 11 And DSA < 16 Then
        a = "X"
        s = DSA - 11
    ElseIf DSA > 7 And DSA < 12 Then
        a = "F"
        s = DSA - 7
    ElseIf DSA > 3 And DSA < 8 Then
        a = "A"
        s = DSA - 3
    Else
        a = "E"
        s = DSA + 1
    End If

    If DC = 15 Then
        c = 20
    Else
        c = DC
    End If
End Sub
```

A.3 MÓDULO (Modulo.bas)

```
'------------------------------------------------------------------------------------------
'Definiciones del puerto de conexión de lector ACSR20
'------------------------------------------------------------------------------------------
Global Const AC_COM1 = 0
'------------------------------------------------------------------------------------------
'Definiciones del tipo de lector (usado en AC_Open)
'------------------------------------------------------------------------------------------
'ACR20 reader (115200 baud)
Global Const ACR20_115200 = &H3
'------------------------------------------------------------------------------------------
'Sección de tarjeta:
'------------------------------------------------------------------------------------------
Global Const AC_AM8KS = 8
'------------------------------------------------------------------------------------------
'Conjunto de comandos ACI
'------------------------------------------------------------------------------------------
Global Const ACI_Read = 1
Global Const ACI_Write = 2
Global Const ACI_SetFuse = 3
Global Const ACI_CardOptions = 3
Global Const ACI_Verify = 4
Global Const ACI_WritePr = 5
Global Const ACI_ChangePIN = 6
Global Const ACI_Erase = 7
Global Const ACI_WriteCarry = 8
Global Const ACI_Authenticate = 9
Global Const ACI_SetProtect = 10
Global Const ACI_ReadProtect = 11
Global Const ACI_SetHE = 12
Global Const ACI_BlowFuse = 10
Global Const ACI_LockProtect = 13
Global Const ACI_ClearProtect = 14
Global Const ACI_WriteAll = 15
Global Const ACI_EraseAll = 16
```

```
'-----------------------------------------------------------------------------------------
'Constantes de Error ACI
'-----------------------------------------------------------------------------------------
Global Const ERR_HANDLE_INVALID = -603
Global Const ERR_SESSION_NULL = -600
Global Const ERR_HANDLE_NOFREE = -108
Global Const ERR_PORT_INVALID = -100
Global Const ERR_READER_INVALID = -101
Global Const ERR_PORT_OCCUPLIED = -102
Global Const ERR_NO_ = -1001
Global Const ERR_NO_CARD = -1002
Global Const ERR_WRONG_ = -1003
Global Const ERR_NO_POWERUP = -1004
Global Const ERR_INVALID_INS = -1005
Global Const ERR_CARD_FAILURE = -1006
Global Const ERR_CARD_PROTOCOL = -1007
Global Const ERR_UNSUPPORT_ = -1008
Global Const ERR_INCOMPATIBLE_COMMAND = -1009
Global Const ERR_ADDRESS = -1010
Global Const ERR_DATA_LEN = -1011
Global Const ERR_RESPONSE_LEN = -1012
Global Const ERR_SECRET_CODE_LOCK = -1013
Global Const ERR_INVALID_MOD_NUMBER = -1014
Global Const ERR_INCORRECT_PASSWORD = -1015
Global Const ERR_INVALID_CLA = -1050
Global Const ERR_APDU_PARAM = -1051
Global Const ERR_COM_BUFFER_FULL = -1052
Global Const ERR_WORD_BOUNDARY = -1053
Global Const ERR_PROTOCOL_FRAME = -1080
Global Const ERR_NO_RESPONSE = -1081
Global Const ERR_INCORRECT_PARAM = -1082
Global Const ERR_UNKNOWN = -1099

Declare Function GetReaderErrorCode Lib "ACSR2032.DLL" () As Integer

'-----------------------------------------------------------------------------------------
'ACR Interfaz
'-----------------------------------------------------------------------------------------
   Type AC_SESSION
      CardType                      As Byte
      SCModule                      As Byte
      ATRLen                        As Byte
      ATR                           As String * 256
      HistLen                       As Byte
      HistOffset                    As Byte
      ApduLenMax                    As Long
   End Type

   Type AC_APDU
      CLA                           As Byte
      INS                           As Byte
      P1                            As Byte
      P2                            As Byte
      Lc                            As Long
      Le                            As Long
      DataIn(1 To 256)              As Byte
      DataOut(1 To 256)             As Byte
      status                        As Long
   End Type

   Type AC_INFO
      nMaxC                         As Long
      nMaxR                         As Long
      CType                         As Long
      CStat                         As Byte
      CSel                          As Byte
      szRev                         As String * 256
      nLibVer                       As Long
      lBaudRate                     As Long
   End Type
```

```
'-----------------------------------------------------------------------------------------
'Sección de prototipo
'-----------------------------------------------------------------------------------------
   Declare Function AC_Open Lib "ACSR2032.DLL" (ByVal ReaderType As Long, ByVal ReaderPort
As Integer) As Long
   Declare Function AC_Close Lib "ACSR2032.DLL" (ByVal hReader As Long) As Long
   Declare Function AC_StartSession Lib "ACSR2032.DLL" (ByVal hReader As Long, Session As
AC_SESSION) As Long
   Declare Function AC_EndSession Lib "ACSR2032.DLL" (ByVal hReader As Long) As Long
   Declare Function AC_GetInfo Lib "ACSR2032.DLL" (ByVal hReader As Long, Info As AC_INFO)
As Long
   Declare Function AC_SetOptions Lib "ACSR2032.DLL" (ByVal hReader As Long, ByVal OType As
Integer, ByVal Value As Integer) As Long
   Declare Function AC_ExchangeAPDU Lib "ACSR2032.DLL" (ByVal hReader As Long, apdu As
AC_APDU) As Long
```

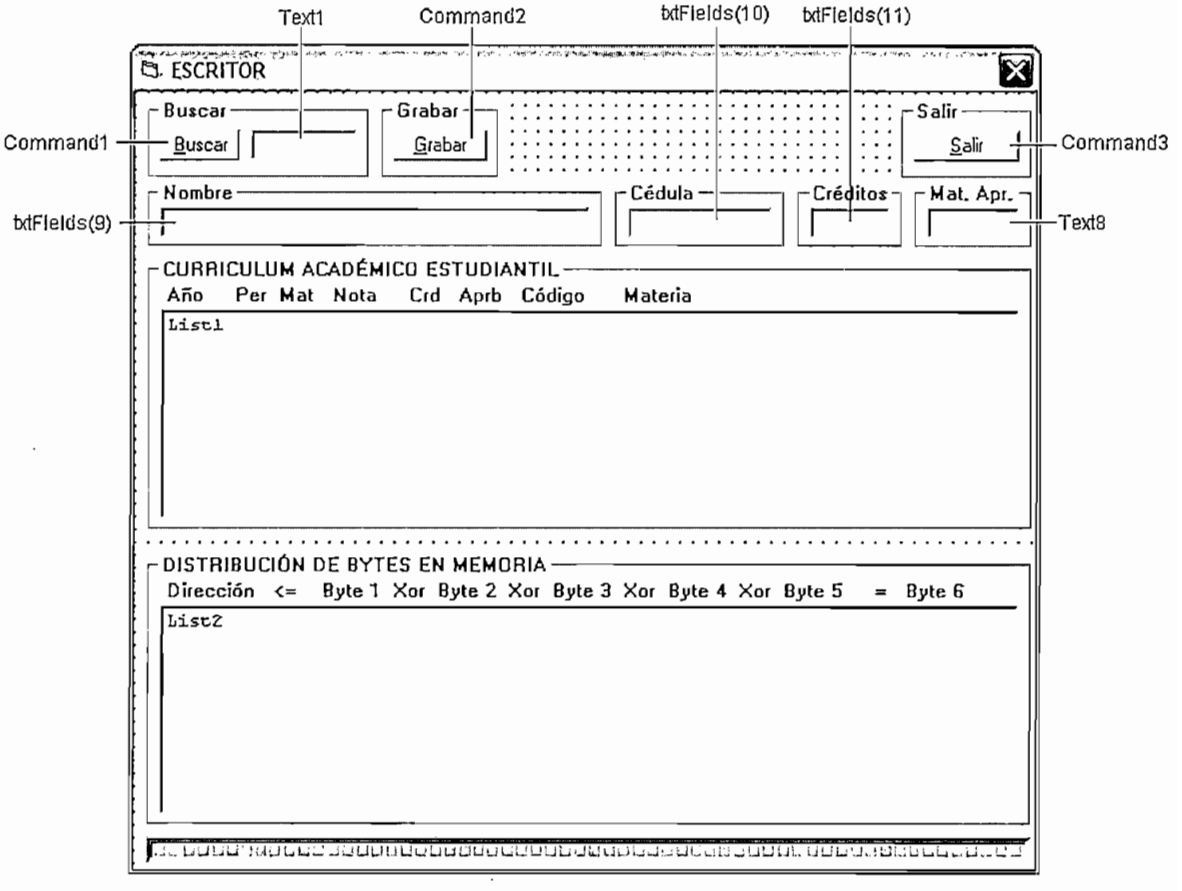

Figura A.1 Formulario de Escritor

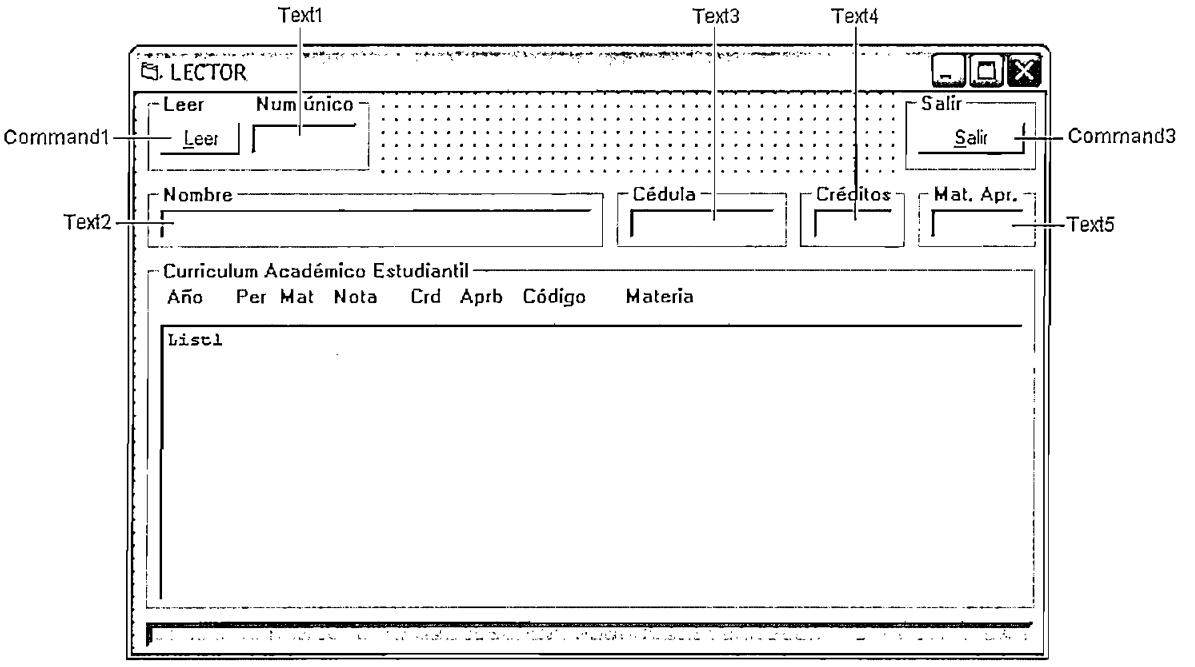

Figura A.2 Formulario de Lector

Anexo B

Tarjeta SLE 4428

B.1 CARACTERÍSTICAS

Las características listadas son aseguradas sobre el rango de operación del circuito integrado. Las características típicas especifican los valores medios esperados sobre la producción difundida. En otro caso no especificado, las características típicas se aplican en TA = 25 ° C y el voltaje dado del suministro.

Las características principales de la tarjeta son las siguientes:

- Organización de EEPROM 1024 x 8 bits
- Direccionamiento de Bytes inteligente
- Protección de escritura inteligente de Bytes irreversible
- Organización de memoria de protección 1024 x 1 bit
- Bus serial de tres vías
- Fin de programación indicada en el dato de salida
- Mínimo de 100000 ciclos escritura/borrado
- Retención de datos: mínimo diez años
- Configuración de contactos e interfaz serial de acuerdo con el estándar 7816 (transmisión sincrónica)
- Los datos solo pueden ser cambiados después de ingresar correctamente los dos Bytes del código de seguridad programable (PSC).

Tabla B.1 Contactos de la tarjeta SLE 4428

Pin	*Contactos de la tarjeta*	*Símbolo*	*Función*
1	C1	VCC	Voltaje de operación 5 V
2	C2	RST	Control del chip
3	C3	CLK	Reloj
4	C4	N.C.	No conectado
5	C5	GND	Tierra
6	C6	N.C.	No conectado
7	C7	E/S	Línea de datos
8	C8	N.C.	No conectado

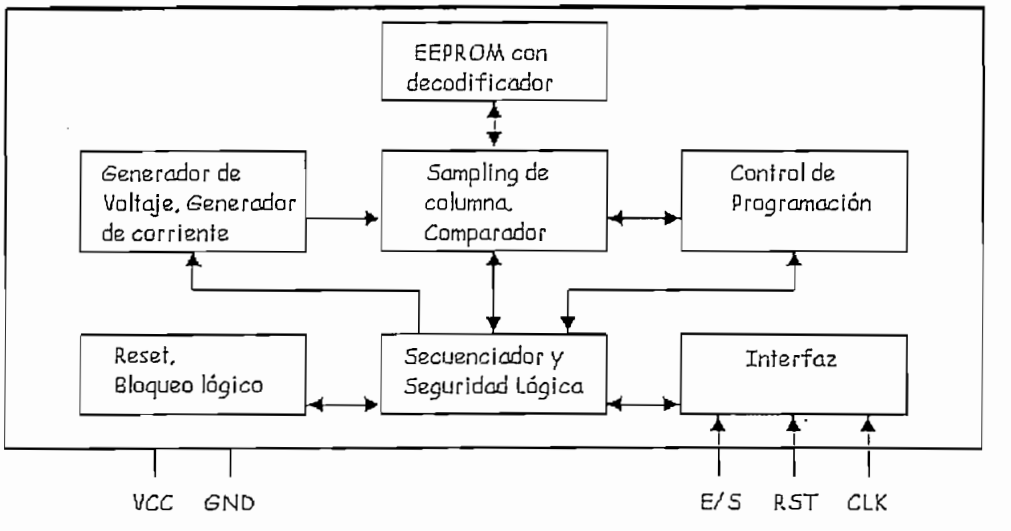

Figura B.1 Diagrama de bloque de operación de SLE 4428

B.2 DESCRIPCIÓN FUNCIONAL

El chip contiene una EEPROM organizada como 1024 x 8 bits ofreciendo la posibilidad de protección de escritura programable para cada byte. Siempre es posible leer la memoria completa. La memoria puede ser escrita y borrada byte por byte. El dato de entrada y el contenido del byte direccionado son comparados de modo que solamente bits que no estaban escritos antes son escritos. El

borrado es posible solo byte por byte, aun si solo se desea borrar un bit, pero los bits pueden ser escritos individualmente. Cada byte puede ser protegido individualmente de escritura/borrado por la puesta de un bit de protección (EEPROM -> ROM). El bit de protección puede ser programado solamente una vez y no puede ser borrado.

Adicionalmente tiene una verificación lógica PSC. Toda la memoria, excepto para el PSC, siempre puede ser leída. La memoria solo puede ser escrita o borrada después de la verificación de PSC. El contador de error siempre puede ser escrito. Después de ocho ingresos incorrectos sucesivos el contador de error bloqueará cualquier subsiguiente intento de verificación de PSC y por lo tanto cualquier posibilidad de escribir o borrar.

B.3 CARACTERÍSTICAS DE SEGURIDAD

Operación extra para identificación de usuario

Sin un ingreso de PSC solo la lectura es posible. El contenido de las direcciones del PSC no puede ser leído. Si se intenta leerla, "00" aparecerá. El procedimiento de verificación del chip debe ser realizado en la siguiente secuencia.

- Escribir un contador de error de no escritura, dirección "1021",
- Ingrese el primer byte del código PSC, dirección "1022",
- Ingrese el segundo byte del código PSC, dirección "1023",
- Después del ingreso correcto el contador de error puede ser borrado.

Después de la verificación de PSC, la línea E/S va de "1" a "0". Este es conmutado a "1" por la transición de RST de "0" a "1". El contador de error no es borrado automáticamente.

Escribiendo el contador de error

Antes de que se ingrese el PSC solo es posible la escritura del contador de error. El número de bits borrados del contador de error determina el número de posibles intentos (max. 8). Después del acceso exitoso al contador de error este debería ser borrado antes de desconectar la fuente de voltaje para reactivar los ocho intentos. Cada error cuando se ingresa el PSC requiere la escritura de un nuevo contador.

Ingreso de PSC

El LSB empezando por el bit menos significante debe ser ingresado primero y luego el más significante. Si la comparación de datos interna prueba ser correcta, la EEPROM está disponible para ser borrada y escrita mientras el voltaje de operación sea aplicado. Después de la habilitación, el PSC puede ser alterado como se desee, excepto los bits de protección que son "0".

B.4 DATOS TÉCNICOS

Tabla B.2 Datos técnicos de la tarjeta SLE 4428

Parámetro	Símbolo	Valores límite Min.	Tip.	Max.	Unidad
Promedio absolutos máximos					
Suministro de voltaje	*Vcc*	-0,3		6	V
Voltaje de entrada	*Vi*	-0,3		6	V
Temperatura almacenada	*Talm*	-40		125	°C
Disipación de Potencia	*Ptot*			60	mW
Rango de Operación					
Suministro de voltaje	*Vcc*	4,5		5,5	V
Temperatura almacenada	*Ta*	-35		100	°C
Características DC					
Suministro					
Suministro de voltaje	*Vcc*	4,5	5	5,5	V
Suministro de corriente	*Icc*	-	3	10	mA
Entrada de datos					
Voltaje de entrada H (E/S,CLK,RST)	*VH*	3,5	-	Vcc	V
Voltaje de entrada L (E/S,CLK,RST)	*VL*	0	-	0,8	V
Corriente de entrada H (E/S,CLK,RST)	*IH*	-	-	10	µA
Salida de datos					
Corriente de salida (VL=0,4V, drenaje abierto)	*IL*	0,5	-	-	mA
Corriente de fuga (VH=VDD drenaje abierto)	*IH*	-	-	10	µA
Capacitancia de entrada	*CI*	-	-	10	pF
Pin de prueba	*T*	abierto o sobre Vss			
Frecuencia de reloj	*fc*	-	20	-	kHz
Duración del pulso					
Tiempo de reset	*tRE*	9	-	-	µs
Answer to reset (RST)	*td6*	20	50	-	µs
CLK (cuenta, nivel H)	*tH*	10	-	-	µs
CLK (cuenta, nivel L)	*tL*	10	-	-	µs
Tiempo de escritura (f=20 kHz)	*tw*	5	-	-	ms
Tiempo de borrado (f=20 kHz)	*tE*	5	-	-	ms
Características AC					
Tiempo de setup (D/CLK)	*td1*	4	-	-	µs
Tiempo de setup (CLK/RST)	*td3*	4	-	-	µs
Tiempo de setup (RST/CLK)	*td4*	4	-	-	µs
Tiempo de retención (D/CLK)	*td5*	4	-	-	µs
Tiempo de retardo (CLK/D)	*td2*	6	-	-	µs
Tiempo de subida (E/S,CLK,RST)	*tR*	-	-	1	µs
Tiempo de caída (E/S,CLK,RST)	*tF*	-	-	1	µs

Anexo C

Librería ACSR 20

En estas líneas se describe el uso del interfaz de software ACSR20 para programar el lector de tarjetas inteligentes ACR30. Esta librería permite que aunque el interfaz de comunicaciones de hardware sea diferente, los programas de aplicación pueden utilizar el mismo API (interfaz de programación de la aplicación) para el funcionamiento de los lectores de tarjetas inteligentes. Realmente, el propósito de usar la librería ACSR20 es proporcionar al programador un interfaz simple y constante sobre todo el hardware posible. Es responsabilidad de la librería ACSR20 manejar los detalles de la comunicación, las conversiones de parámetros y la gestión de error. La arquitectura de la librería ACSR20 se puede visualizar como el diagrama de la figura C.1

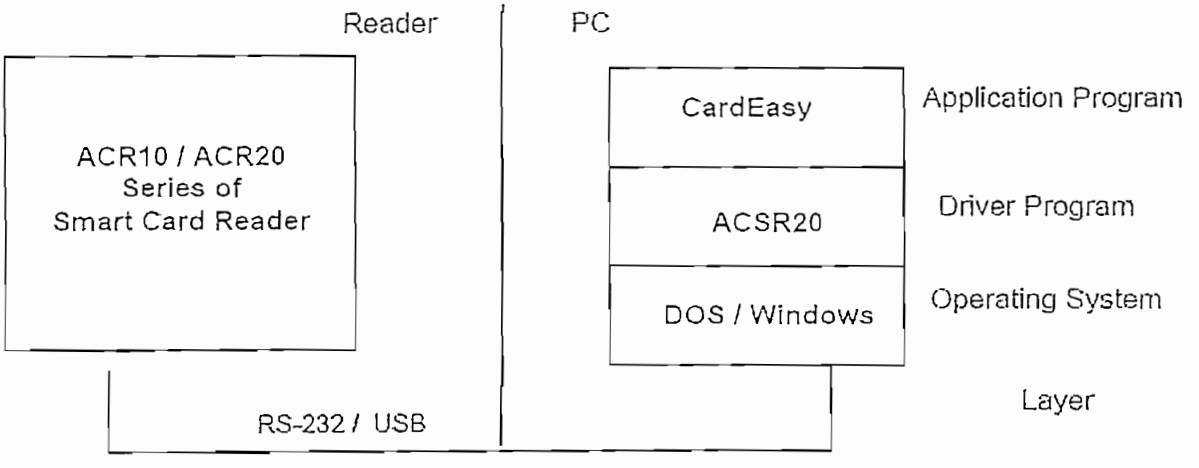

Figura C.1 Arquitectura de la librería ACSR20

C.1 DESCRIPCIÓN

ACSR20 es un conjunto de funciones de alto nivel proporcionadas para el uso del software de aplicación. Esta proporciona un Interfaz de Programación de

Aplicación consistente (ACSR20 API) para que la aplicación opere sobre el lector de tarjetas y la correspondiente tarjeta insertada. ACSR20 se comunica con el lector ACR30 vía las facilidades del puerto de comunicación proporcionadas por el sistema operativo. ACSR20 se ha diseñado para ser independiente de la plataforma proveída de allí que se requiere una mínima modificación sobre el módulo de comunicación del ACSR20 para adaptarse a diferentes ambientes de funcionamiento.

C.2 VELOCIDAD DE COMUNICACIÓN

La librería ACSR20 controla la velocidad de comunicación entre el lector y el PC. Para aquellos lectores que usan la conexión serial RS232, la tasa de baudios de comunicación por defecto (ajustes de fabricación) es 9600 bps, sin paridad, 8 bits de datos y un bit de parada. Una velocidad más alta de 115200 bps puede ser activada a través del uso de un comando de software emitido por el Host. Si no esta seguro de los ajustes de fabricación de su lector, siempre puede usar la opción ACR_AUTODETECT en su comando AC_Open para que el Driver detecte la velocidad de comunicación automáticamente.

C.3 API ACSR

El API ACSR20 define una vía común de acceso al lector ACR30. Los programas de Aplicación invocan a ACSR20 a través de las funciones de interfaz y realizan operaciones sobre la tarjeta insertada a través del uso de comandos ACI. El archivo de cabecera ACSR20 está disponible para el desarrollador de programas con el contenido de las funciones de prototipo y macros descritas a continuación:

C.3.1 ESTRUCTURA DEL INTERFAZ DE DATOS

El API ACSR20 hace uso de algunas estructuras de datos para pasar parámetros entre el programa de aplicación y el driver de la biblioteca. Estas estructuras de datos son definidas en el archivo de cabecera ACSR20 y se discuten a continuación:

C.3.1.1 AC_APDU

La forma de definir la estructura es la siguiente:

```
Type AC_APDU
        CLA                 As Byte
        INS                 As Byte
        P1                  As Byte
        P2                  As Byte
        Lc                  As Long
        Le                  As Long
        DataIn(1 To 256)    As Byte
        DataOut(1 To 256)   As Byte
        Status              As Long
End Type
```

La estructura de datos AC_APDU se usa en la función AC_ExchangeAPDU para el paso de comandos y de datos al interior de la tarjeta. Se debe tener el manual de referencia de la tarjeta a utilizar antes de realizar cualquier operación valida en la tarjeta. Por favor note que **Lc** representa la longitud de los datos que

van a entrar en la tarjeta y Le representa la longitud de los datos que se espera recibir de la tarjeta.

Tabla C.1 Parámetros de la estructura AC_APDU

Nombre	*Entrada/Salida*	*Descripción*
CLA	E	Instrucción de clase
INS	E	Instrucción de código
P1	E	Parámetro 1
P2	E	Parámetro 1
Lc	E	Longitud de los datos del comando
Le	E/S	Longitud de los datos de la respuesta
DataIn	E	Buffer de datos de comando
DataOut	S	Buffer de datos de respuesta
Status	S	Estado de ejecución del comando

C.3.1.2 AC_SESSION

La forma de definir la estructura es la siguiente:

```
Type AC_SESSION
        CardType        As Byte
        SCModule        As Byte
        ATRLen          As Byte
        ATR             As String * 256
        HistLen         As Byte
        HistOffset      As Byte
        ApduLenMax      As Long
End Type
```

La estructura de datos AC_SESSION se usa en la función AC_StartSession, llamada para la recuperación de información ATR de la TI. Antes de llamar a AC_ StartSession, el programa necesita especificar los valores de CardType. Después de llamar la función, la cadena ATR puede ser encontrada en el campo ATR y la longitud es almacenada en ATRLen.

Tabla C.2 Parámetros de la estructura AC_SESSION

Nombre	*Entrada/Salida*	*Descripción*
CardType	E	El tipo de tarjeta seleccionado para operación
SCModule	E	El módulo de seguridad seleccionado para la operación
ATRLen	S	Longitud de la cadena ATR
ATR	S	Atención al cadena reset (ATR)
HistLen	S	No se usa - campo obsoleto
HistOffset	S	No se usa - campo obsoleto
APDULenMax	S	No se usa - campo obsoleto

C.3.1.3 AC_INFO

La forma de definir la estructura es la siguiente:

```
Type AC_INFO
        nMaxC          As Long
        nMaxR          As Long
        CType          As Long
        CStat          As Byte
        CSel           As Byte
        SzRev          As String * 256
        NLibVer        As Long
        LBaudRate      As Long
End Type
```

Tabla C.3 Parámetros de la estructura AC_INFO

Nombre	*Entrada/Salida*	*Descripción*
nMaxC	E	El número máximo de bytes de datos del comando (DataIn) que pueden ser aceptados en el comando ExchangeAPDU
nMaxR	E	El número máximo de bytes de datos de la respuesta (DataOut) que podrían hacerce presentes en el comando ExchangeAPDU
CType	E	Los tipos de tarjetas soportados por el lector
Cstat	E	El estado del lector Bit0= tarjeta presente (1) o ausente (0) Bit1= tarjeta energizada (1) o no (0)
szRev[10]	E	El código de revisión firmware
nLibVer	E	La versión de la librería (e.j. 310 es igual a versión 3,10)

La estructura de datos AC_INFO se usa en la función AC_GetInfo llamada para la recuperación de la información relacionada con el lector. Su significado se describe en la tabla C.3

C.3.2 PROTOTIPOS DE FUNCIONES DE INTERFAZ

Generalmente, un programa requiere antes que nada llamar a AC_open para obtener una manija. La manija se requiere para subsecuentes llamadas a AC_StartSession, AC_ExchangeAPDU, AC_EndSession y AC_Close. La tarjeta insertada puede ser energizada usando la función AC_StartSession y los comandos pueden ser intercambiados con la tarjeta insertada usando la función AC_ExchangeAPDU. Por otra parte, AC_SetOptions y AC_GetInfo son dos comandos que pueden ser usados para poner y leer información variada del lector

C.3.2.1 AC_Open

Esta función abre un puerto y retorna una manija de lector valida para el programa de aplicación.

Formato:

Declare Function AC_Open Lib "ACSR2032.DLL" (ByVal ReaderType As Long, ByVal ReaderPort As Integer) As Long

Parámetros de entrada:

La tabla C.4 lista los parámetros para esta función.

Respuesta:

El valor de respuesta es negativo y contiene el código de error cuando la función encuentra un error durante su operación. En otro caso, esta devuelve una manija de lector valida. Favor referirse a la Tabla C.18 para la descripción en detalle y el significado de los códigos de error.

Tabla C.4 Parámetros de la función AC_Open

Parámetros	*Definición/Valores*
ReaderType	Tipo de lector y velocidad de comunicación ACR10= El lector seleccionado es ACR10 (9600bps) ACR20= El lector seleccionado es ACR20 (57600bps) ACR20_57600= El lector seleccionado es ACR20 (57600bps) ACR20_19200= El lector seleccionado es ACR20 (19200bps) ACR20_115200= El lector seleccionado es ACR20 (115200bps) ACR20_9600= El lector seleccionado es ACR20 (9600bps) ACR_AUTODETECT = Autodetectar el lector
RearerPort	Puerto en el que está conectado el lector AC_COM1=Puerto de comunicaciones estándar 1 AC_COM2=Puerto de comunicaciones estándar 2 AC_COM3=Puerto de comunicaciones estándar 3 (Válido solo en el driver Win32) AC_COM4=Puerto de comunicaciones estándar 4 (Válido solo en el driver Win32) AC_USB=Usando el puerto de comunicaciones USB AC_PCSC=Usando el componente PCSC para acceder al lector

Observaciones:

Si el usuario selecciona ACR_AUTODETECT para abrir un lector la primera vez, la biblioteca tratará de resetear el lector y espera el mensaje de respuesta del lector. Para que la biblioteca complete esta operación podría tomar algún tiempo (alrededor de 2-3 segundos).

C.3.2.2 AC_Close

Esta función cierra un puerto de lector abierto previamente.

Formato:

Declare Function AC_Close Lib "ACSR2032.DLL" (ByVal hReader As Long) As Long

Parámetros de entrada:

Tabla C.5 Parámetros de la función AC_Close

Parámetros	*Definición/Valores*
hReader	Una manija de lector válida previamente abierta con AC_Open()

La tabla C.5 lista los parámetros de entrada de esta función.

Respuesta:

El valor de la respuesta es cero si la función es satisfactoria. En otro caso, esta retorna un valor negativo conteniendo el código de error. Para el significado detallado del código de error, favor referirse a la tabla C.18.

C.3.2.3 AC_StartSession

Esta función inicia una sesión con un tipo de tarjeta seleccionado y actualiza la estructura **sesión** con los valores devueltos por el Answer-.to-Reset (ATR) de la tarjeta. Una sesión se inicia con un reset de la tarjeta y se finaliza ya sea con otro reset de la tarjeta, al quitarle la energía a la tarjeta o al removerla del lector. Note que esta función energizará la tarjeta y realizará un reset en ella.

Formato:

Declare Function AC_StartSession Lib "ACSR2032.DLL" (ByVal hReader As Long, Session As AC_SESSION) As Long

Parámetros de entrada:

La tabla C.6 lista los parámetros de entrada de esta función.

Tabla C.6 Parámetros de entrada de la función AC_StartSession

Parámetros	*Definición/Valores*
hReader	Una manija de lector válida previamente abierta con AC_Open()
Session.CardType	El tipo de tarjeta seleccionado para esta sesión
Session.SCModule	El número de módulo de seguridad seleccionado (requerido solo cuando tipo de tarjeta =AC_SCModule)

Parámetros de salida:

La tabla C.7 lista los parámetros de salida de esta función.

Tabla C.7 Parámetros de salida de la función AC_StartSession

Parámetros	*Definición/Valores*
Session.ATR	Answer to reset retornado por la tarjeta
Session.ATRLen	Longitud del ATR
Session.HistLen	Longitud de datos históricos
Session.HistOffset	Offset de los datos históricos
Session.APDULenMax	Longitud máxima de APDU soportada

Respuesta:

El valor devuelto es cero si la función se realizó satisfactoriamente. En otro caso, esta devuelve un valor negativo conteniendo el código de error. Para el significado detallado del código de error, favor referirse a la tabla C.18.

Observaciones:

Cuando ningún tipo de tarjeta es seleccionado (es decir, Session.CardType=0), el lector tratará de detectar automáticamente el tipo de tarjeta insertada. Sin embargo, mientras el lector puede distinguir tarjetas T=0, T=1 y Memory cards sincrónicas, este no puede distinguir los diferentes tipos de Memory Cards.

C.3.2.4 AC_EndSession

Esta función finaliza una sesión iniciada previamente y le quita la fuente de energía a la tarjeta.

Formato:

Declare Function AC_EndSession Lib "ACSR2032.DLL" (ByVal hReader As Long) As Long

Parámetros de entrada:

La tabla de abajo lista los parámetros de entrada de esta función.

Tabla C.8 Parámetros de entrada de la función AC_EndSession

Parámetros	*Definición/Valores*
hReader	Una manija de lector válida previamente abierta con AC_Open()

Respuesta:

El valor devuelto es cero si la función se realiza satisfactoriamente, en otro caso, esta devuelve un valor negativo conteniendo el código de error. Para el significado detallado del código de error, favor referirse a la Tabla C.18.

C.3.2.5 AC_Exchange

Esta función envía un comando APDU para una tarjeta vía el puerto abierto y retorna la respuesta de la tarjeta. Favor referirse a la sección 2.3.3 Comandos ACI para una descripción en detalle sobre como llenar estos parámetros.

Formato:

Declare Function AC_ExchangeAPDU Lib "ACSR2032.DLL" (ByVal hReader As Long, Apdu As AC_APDU) As Long

Parámetros de entrada:

La tabla C.9 lista los parámetros de entrada de esta función.

Tabla C.9 Parámetros de entrada de la función AC_Exchange

Parámetros	*Definición/Valores*
hReader	Una manija válida retornada por AC_Open()
Apdu.CLA	Instrucción de clase
Apdu.INS	Instrucción de código
Apdu.P1	Parámetro 1
Apdu.P2	Parámetro 1
Apdu.DataIn	Buffer de datos a ser enviados
Apdu.Lc	Número de Bytes en Apdu.DataIn a ser enviados
Apdu.Le	Número de Bytes que se espera recibir

Parámetros de salida:

La tabla C.10 lista los parámetros de salida de esta función.

Tabla C.10 Parámetros de salida de la función AC_Exchange

Parámetros	*Definición/Valores*
Apdu.DataOut	Buffer de datos que contienen la respuesta de la tarjeta
Apdu.Le	Número de Bytes recibidos en Apdu.DataOut
Apdu.Status	Bytes de estado SW1, SW2 retornado por la tarjeta

Respuesta:

El valor devuelto es cero si la función se realiza satisfactoriamente. En otro caso, esta devuelve un valor negativo conteniendo el código de error. Para el significado detallado del código de error, favor referirse a la Tabla C.18.

C.3.2.6 AC_GetInfo

Esta función recupera información relacionada con el lector seleccionado en ese momento.

Formato:

Declare Function AC_GetInfo Lib "ACSR2032.DLL" (ByVal hReader As Long, Info As AC_INFO) As Long

Parámetros de entrada:

La tabla C.11 lista los parámetros de entrada de esta función.

Tabla C.11 Parámetros de entrada de la función AC_GetInfo

Parámetros	*Definición/Valores*
hReader	Una manija de lector válida previamente abierta con AC_Open()

Parámetros de salida:

La tabla C.12 lista los parámetros de salida de esta función.

Tabla C.12 Parámetros de salida de la función AC_GetInfo

Parámetros	*Definición/Valores*
Info.szRev	Código de revisión del lector seleccionado
Info.nMaxC	El número máximo de bytes de datos del comando
Info.nMaxR	El número máximo de bytes de datos que pueden ser recibidos al ser transmitidos en una respuesta
Info.CType	Los tipos de tarjetas soportados por este lector
Info.CStat	El estado actual del lector 0 = ninguna tarjeta insertada 1 = tarjeta insertada, no energizada 3 = tarjeta energizada
Info.CSel	El tipo de tarjeta seleccionado actualmente
Info.nLibVer	Versión de librería actual
Info.lBaudRate	La tasa de transmisión actualmente en uso

Respuesta:

El valor devuelto es cero si la función se realiza satisfactoriamente. En otro caso, esta devuelve un valor negativo conteniendo el código de error. Para el significado detallado del código de error, favor referirse a la Tabla C.18.

C.3.3 COMANDOS ACI

Los comandos ACI se proporcionan para soportar la operación estándar de un amplio rango de memory cards. Porque de la diferente naturaleza de diferentes memory cards y sus propiedades, no todos los comandos están disponibles para cada tipo de tarjeta. El apéndice B lista los comandos ACI disponibles para diferentes tipos de tarjetas.

Los comandos ACI descritos aquí son usados en llamadas a la función AC_ExchangeAPDU. Valores apropiados de parámetros de diferentes comandos ACI podrían ser inicializados en la estructura AC_APDU antes de llamar a la función AC_ExchangeAPDU. Para que la función se complete satisfactoriamente, el programa de aplicación requiere chequear el estado devuelto por la tarjeta (SW1 y SW2) en AC_APDU y por consiguiente recuperar los resultados de salida desde el DataOut buffer.

C.3.3.1 ACI_Read

Este comando esta disponible para todos los tipos de Memory Cards. La estructura de parámetros se define como sigue:

Tabla C.13 Estructura de parámetros del comando ACI_Read

Campo	*Valor*	*Descripción*
CLA	0x00	Instrucción de clase
INS	ACI_Read	Instrucción del código para el comando Leer
P1	Variable	MSB de la dirección de inicio a ser leída
P2	Variable	LSB de la dirección de inicio a ser leída
Lc	0x00	Ningún dato de entrada es requerido para este comando
DataIn	-	Sin cuidado
Le	Variable	El número de bytes a ser leídos de la tarjeta

C.3.3.2 ACI_Write

Este comando esta disponible para lodos los tipos de Memory Cards. La estructura de parámetros se define como sigue:

Tabla C.14 Estructura de parámetros del comando ACI_Write

Campo	*Valor*	*Descripción*
CLA	0x00	Instrucción de clase
INS	ACI_Write	Instrucción del código para el comando Escribir
P1	Variable	MSB de la dirección de inicio a ser escrita
P2	Variable	LSB de la dirección de inicio a ser escrita
Lc	Variable	El número de bytes a ser escritos en la tarjeta
DataIn	Variable	Lc Bytes de datos a ser escritos en la tarjeta
Le	0x00	Ninguna respuesta esperada de la tarjeta

C.3.3.2 ACI_Verify

Este comando esta disponible para algunas memory cards con la característica de código secreto. Esto no es suficiente para chequear solo el código devuelto desde la función AC_ExchangeAPDU para determinar si la verificación es o no satisfactoria. El programa de aplicación deberá chequeare el dato devuelto desde la tarjeta para la interpretación de si la verificación es o no satisfactoria.

Tabla C.15 Estructura de parámetros del comando ACI_Verify

Campo	*Valor*	*Descripción*
CLA	0x00	Instrucción de clase
INS	ACI_Verify	Instrucción del código para el comando Verificar
P1	Variable	Índice de llave, ver tabla C.16 para la descripción
P2	-	Sin cuidado
Lc	Variable	El número de bytes de la llave
DataIn	Variable	Lc Bytes de valor de la llave para ser verificados en la tarjeta
Le	Variable	Caso cuando: AC_AM2KS : Le = 4 AC_AM8KS : Le = 3 En otro caso : Le = 0

Dependiendo del tipo de tarjeta de la sesión que se este en uso, Key Index (P1) puede contener alguno de los siguientes valores.

Tabla C.16 Índice de llave

Tipo de tarjeta	*Indice de la llave*	*Longitud (Bytes)*	*Descripción*
AC_AM104	0x00	3	Código de transporte
AC_AM221	0x00	3	Código de transporte
AC_SLE4404	0x00	2	Código de seguridad (GPM416)/código de seguridad (SLE4404)
	0x01	4	Código de borrado (GPM416)/ código de sistema (SLE4404)
AC_GPM896	0x00	2	Código de seguridad de la tarjeta
	0x01	6	Código de borrado 1
	0x02	4	Código de borrado 2
AC_T101	0x00	2	Código de seguridad
	0x01	4	Llave de borrado zona 1
AC_T102	0x00	2	Código de seguridad
	0x01	6	Llave de borrado zona 1
	0x02	4	Llave de borrado zona 2
AC_AT1604	0x00	2	Código de seguridad
	0x01	2	Llave de borrado zona 1
	0x02	2	Llave de borrado zona 2
	0x03	2	Llave de borrado zona 3
	0x04	2	Llave de borrado zona 4
	0x05	2	Código de seguridad Zona 1
	0x06	2	Código de seguridad Zona 2
	0x07	2	Código de seguridad Zona 3
	0x08	2	Código de seguridad Zona 4
AC_AM2KS	0x00	3	Código de seguridad
AC_AM8KS	0x00	2	Código de seguridad
AC_X76F041	0x00	8	Password de escritura
	0x01	8	Password de lectura
	0x02	8	Password de configuración

AC_X76F128 O AC_X76F640	0x00	8	Password leer vector 0
AC_X76F100	0x01	8	Password de escritura
	0x02	8	Password de lectura

Tabla C.17 Compatibilidad de tipos de tarjetas

ACSR20 card type	ACS	Atmel	Gemplus	SGSThomson	Siemens	Xicor
AC_AM104	AM104	AT88SC06	GPM103	ST1305	SLE4406	
AC_AM221	AM221	-	GAM226	-	SLE4436	-
AC_SLE4404	AM416	-	GPM416	-	SLE4404	-
AC_GPM896	-	-	GPM896	-	-	-
AC_AT101	-	AT88SC101	-	-	-	-
AC_AT102	-	AT88SC102	-	-	-	-
AC_AM8KP	AM8KP	-	-	-	SLE4418	-
AC_AM8KS	AM8KS	-	-	-	SLE4428	-
AC_AM2KP	AM2KP	-	-	-	SLE4432	-
AC_AM2KS	AM2KS	-	-	-	SLE4442	-
AC_IIC	AM1KF	AT24C01	GFM1K	ST14C02C		X24026
	AM2KF	AT24C02	GFM2K	ST14C04C		X24165
	AM4KF	AT24C04	GFM4K			X24645
		AT24C08	GFM8K			
		AT24C16				
AC_XIIC	AM64KP	-	-	-	-	-
AC_AT1604	-	AT88SC1604	-	-	-	-
AC_AM256	AM256	-	-	-	-	-
AC_AM4KP	AM4KP	-	-	-	-	-
AC_X76F041	-	-	-	-	-	X76F041
AC_X24645	-	-	-	-	-	X24645
AC_ST1335	-	-	-	ST1335	-	-
AC_ST1333	-	-	-	ST1333	-	-
AC_X76F128	-	-	-	-	-	X76F128
AC_X76F640	-	-	-	-	-	X76F640
AC_X76F100	-	-	-	-	-	X76F100
AC_T0	AllT=0MCUcard					
AC_T1	AllT=1MCUcard					
AC_SCModule	AllT=0/T=1securitymoduleinSIMMform					

Tabla C.18 Códigos de Error

Código	*Significado*
-603	Error en el manejo del lector
-600	Parámetros de Sesión nulo
-108	Ninguna manija libre para la fijación
-100	Puerto seleccionado no es valido
-101	Lector seleccionado no es valido
-102	Puerto seleccionado está ocupado
-1001	No se ha seleccionado un tipo de tarjeta
-1002	La tarjeta no ha sido insertada
-1003	Tipo de tarjeta errado
-1004	Tarjeta no energizada
-1005	INS no es valido
-1006	Falla en la tarjeta
-1007	Error en el protocolo
-1008	Tipo de tarjeta no soportado
-1009	Comando incompatible
-1010	Error en dirección
-1011	Error en longitud de los datos
-1012	Error en longitud de la respuesta
-1013	Código secreto bloqueado
-1014	Número de modulo SC no valido
-1015	Password incorrecto
-1050	Error en CLA
-1051	Error en parámetros APDU
-1052	El Buffer de comunicación está lleno
-1053	Direcciones no alineadas con palabra límite
-1080	Error en la trama del protocolo
-1081	Ninguna respuesta del lector
-1082	Error encontrado al llamar a los parámetros de la función
-1083	Funciones especificadas no soportadas
-1084	Circuito corto de conector
-1085	Error interno inesperado
-1086	Falta un archivo DLL requerido
-1099	Respuesta desconocida
-2000	Error USB interno
-2001	Error en localidad de memoria
-2002	Error en enlace de la librería USB
-2003	Error en localizar directorio de sistema de ventana
-3000	Error encontrado en administrador PCSC smart card

Anexo D

Lector y Programador ACR30

D.1 INTRODUCCIÓN

El lector y programador de TIs ACR30 está conectado al computador a través de un interfaz serial asincrónico (RS-232). El lector acepta comandos de la computadora, traslada la función especificada a la TI y devuelve los datos de respuesta o la información de estado.

Las características principales del lector son las siguientes:

- Interfaz de TIs compatible con ISO 7816 - 1/2/3
- Soporta tarjetas con CPU con protocolos T=0 y T=1
- Soporta Memory cards comunmente usadas (I2C, SLE4406, SLE4418/28, SLE4432/42)
- Soporta PPS (Selección de protocolos y parámetros) con 9600 – 96000 bps al leer y escribir en tarjetas inteligentes.
- Interfaz RS – 232 al PC con una estructura de comando simple.

D.2 TIPOS DE TARJETAS SOPORTADOS

El ACR30 puede operar tarjetas MCU con el protocolo T=0 o T=1. La Tabla D.1 explica cual es el valor que se debe ingresar para seleccionar una tarjeta de entre las distintas soportadas.

D.2.1 MEMORY CARDS (INTERFAZ SINCRÓNICA)

- Tarjetas con EEPROM tipo '104' con testigo de conteo no recargable, incluye:
  - Gemplus GPM103
  - Siemens
- Tarjeta Siemens SLE4432/4442 inteligente 256 bytes EEPROM con función de protección de escritura.
- Tarjeta Siemens SLE4418/4428 inteligente 1K bytes EEPROM con función de protección de escritura.

D.2.2 TARJETAS BASADAS EN MICROCONTROLADORES (INTERFAZ ASINCRÓNICA)

El ACR30 soporta tarjetas basadas en microcontroladores con generación de voltaje interno programado (Vpp) y los siguientes parámetros transmitidos en el ATR:

PI = 0 a 5

I = 25 a 50

El ACR30 realiza el procedimiento de selección de protocolo y parámetros (PPS) como se especifica en ISO7816-3

D.3 INTERFAZ DE LA TARJETA INTELIGENTE

La interfaz entre el ACR30 y la tarjeta inteligente insertada sigue las especificaciones de ISO7816-3 con ciertas restricciones o mejoras para incrementar la funcionalidad práctica del ACR30.

D.3.1 SUMINISTRO DE ENERGÍA A LA TARJETA INTELIGENTE (VCC)

El consumo de corriente de la tarjeta insertada no debe ser mayor que 50 mA.

D.3.2 VOLTAJE DE PROGRAMACIÓN (VPP)

De acuerdo con ISO 7816-3, el contacto C6 (VPP) de la tarjeta inteligente provee el voltaje de programación a la tarjeta inteligente. Dado que todas las tarjetas inteligentes comunes en el mercado son basadas en EEPROM y no requieren la provisión de un voltaje de programación externo, el contacto C6 (VPP) ha sido implementado como una señal de control en el ACR30. Las especificaciones eléctricas de este contacto son idénticas a aquellas de la señal RST (contacto C2).

D.3.3 SELECCIÓN DEL TIPO DE TARJETA

El PC que tiene el control siempre debe seleccionar el tipo de tarjeta a través del envió del comando adecuado para que el ACR30 active la tarjeta insertada. Esto incluye tanto Memory cards como tarjetas basadas en CPU.

Para tarjetas basadas en microcontroladores el lector permite seleccionar el protocolo preferido, T=0 o T=1. Sin embargo, esta selección solo es aceptada y trasladada por el lector a través de un PPS cuando la tarjeta insertada en el lector soporta ambos tipos de protocolos. Cuando una tarjeta basada en microcontrolador soporta solo un protocolo, T=0 o T=1, el lector automáticamente usa ese tipo de protocolo, independientemente del tipo de protocolo seleccionado por la aplicación.

D.3.4 INTERFAZ PARA TARJETAS BASADAS EN MICROCONTROLADOR

Para tarjetas basadas en microcontrolador solo los contactos C1 (VCC), C2 (RST), C3 (CLK), C5 (GND) y C7 (E/S) son usados. Una frecuencia de 3.6864 / 4 MHz es aplicada a la señal de reloj (C3).

D.3.5 PROTECCION DE DESPLAZAMIENTO DE LA TARJETA

El ACR30 provee un mecanismo de protección al insertar la tarjeta cuando está es súbitamente retraída mientras es energizada. La energía suministrada a la tarjeta y las líneas de señal entre el ACR30 y la tarjeta son inmediatamente desactivadas cuando la tarjeta está siendo removida. Como una regla general, como sea, para evitar el daño eléctrico, una tarjeta debería ser removida del lector únicamente mientras esta no está energizada.

D.4 SUMINISTRO DE ENERGÍA

El ACR30 requiere un voltaje de 5V DC, 100 mA, regulado, suministro de energía. El ACR30 obtiene el suministro de energía del PC (a través del cable suministrado con cada tipo de lector).

Leds de estado

El led verde en el lado frontal del lector indica el estado de activación del interfaz de la TI.

D.5 INTERFAZ SERIAL

El ACR30 está conectado a la computadora a través de una interfaz serial asíncrona siguiendo el estándar RS-232.

D.5.1 PARÁMETROS DE COMUNICACIÓN

Los siguientes parámetros de comunicación son usados por el ACR30 y no puede ser modificada por la computadora host.

Protocolo de transmisión:	serial asincrónico
Paridad:	ninguna
Bits de datos:	8
Bits de parada:	1
Handshake:	a través de CTS

El ACR30 provee dos formas principales de seleccionar la velocidad de transmisión (tasa de baudios) usados por el lector en la operación normal, por hardware y/o por software.

D.6 PROTOCOLO DE COMUNICACIÓN

En la operación normal, el ACR30 actúa como un dispositivo esclavo con respecto a la comunicación entre la computadora y el lector. La comunicación es llevada a cavo en la forma de sucesivos intercambios de comandos-respuestas. La computadora trasmite un comando al lector y recibe una respuesta del lector después de que el comando ha sido ejecutado. Un nuevo comando puede ser transmitido al ACR30 solo después de que la respuesta del comando previo ha sido recibida.