Università degli Studi di Camerino

Scuola di Scienze e Tecnologie

Corso di Laurea in Informatica

GUI smart card file manager: Enjoy My Unicam Card

Laureando:
Edmondo Maria Barocci
matricola 078426

Relatore:
Fausto Marcantoni

Anno Accademico 2013/2014

1. Introduzione

L'evoluzione delle nuove tecnologie ed in particolare la miniaturizzazione dell'hardware, hanno permesso di creare strumenti di grande utilità, come le Smart-Card, che in pochi millimetri di spessore racchiudono la tecnologia per memorizzare, comunicare e calcolare.

L'ambito d'uso per queste tessere è molteplice, e sarebbe riduttivo farne un elenco puntato; basti pensare alle capacità di memorizzazione di dati personali ed al concetto di identificazione (non solo nelle transazioni online) che ne viene, oppure all'utilizzo in ambito bancario ed al settore sia pubblico che privato; in potenza potrebbero sostituire qualsiasi certificato o documento personale: carta d'identità, patente di guida, passaporto, certificato elettorale, certificato medico, storia medica e delle allergie, abilitazioni varie, e molto altro. Essendo alcune Smart-Card dotate non solo di memory card ma anche di un microprocessore, sono una base salda come piattaforma crittografica per l'identificazione telematica.

Nella prima parte di questa tesi verranno discusse le tecnologie alla base dello scambio di informazioni fra una tessera con chip e un lettore[1] di Smart-Card, mentre nella seconda parte verrà presentata un implementazione delle modalità e degli standard di utilizzo, nella forma di un programma applicativo. La funzione del software è quella di permettere all'utente di inserire dati di qualsiasi tipo, sotto forma di file di testo, nel sistema gerarchico di directory memorizzato nella tessera, tramite l'utilizzo di comandi specifici di lettura e scrittura; il livello di astrazione è massimo in quanto l'utilizzatore ha a che fare solo con nomi e testo e non con comandi composti da sequenze di byte.

1 Verrà chiamato lettore per coerenza con le fonti, nonostante operi sia in lettura che scrittura

1.1. Storia delle Smart-Card

La storia trentennale delle Smart-card comincia con i chip integrati in tessere, nel 1968 con gli ingegneri Jürgen Dethloff ed Helmut Grötrupp che depositano i primi brevetti di circuito integrato, seguiti poi dal Dr. Kunitaka Arimura che nel 1970, deposita un brevetto del concetto di Smart-Card. Più tardi, nel 1974 l'inventore francese Roland Moreno deposita un brevetto della prima tessera a circuito integrato ICC, che sarà poi definita Smart-Card; tre anni dopo le aziende Bull CP8, SGS Thomson, e Schlumberger iniziano la produzione in scala delle tessere, e nel 1979 Motorola sviluppa il primo microcontrollore per chip che verrà utilizzato alcuni anni dopo nel circuito bancario francese. Sempre in Francia avviene il primo test sul campo per le tessere elettroniche in un utilizzo massivo per le linee telefoniche a pagamento a partire dal 1982; negli anni seguenti lo sviluppo e la diffusione proseguono espandendosi in Europa ed approdando negli Stati Uniti dove l'utilizzo delle Smart-Card viene implementato in larga scala, in varie banche, aziende e organi governativi, per tutto il corso degli anni ottanta, nei numeri di decine di migliaia di tessere. Un ulteriore passo importante è quello del 1994 quando le realtà “locali” non sono più gestibili singolarmente ed Europay, MasterCard e Visa (EMV) pubblicano delle specifiche congiunte, a livello globale, per tessere bancarie con microchip. Negli anni successivi in Germania vengono emesse ottanta milioni di schede con chip di memoria che svolgono la funzione di tessera sanitaria ed in Francia, alcuni anni dopo viene lanciato lo stesso servizio. Lo sviluppo globale prosegue poi fino ai giorni nostri in cui si fa un uso estensivo delle Smart-Card che possono essere intercompatibili anche fra diverse nazioni e che offrono molteplici funzioni: l'esempio maggiore è quello della carta d'identità malese MyKad che comprende otto applicazioni e svariate funzionalità, ed è utilizzata da 18 milioni di utenti[17][18][19].

2. Hardware

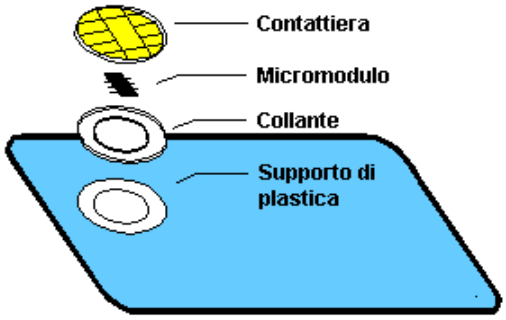

Figura 1: schema smart-card

Una Smart-card è una tessera di materiale plastico, con un chip integrato.

Su base fisica, si presenta come una carta di 85.60 per 53.98 millimetri, con uno spessore di 0.8 millimetri, ed una placca d'oro per facilitare la trasmissione elettrica, d' informazioni in andata e ritorno da ed alla carta. Accoppiato ad essa si usa un lettore, che è un device con una piastra di appoggio che invia e riceve segnali elettro-magnetici, e con un ingresso per la lettura ti tessere a contatto. Le caratteristiche fisiche generali rispecchiano lo standard ISO/IEC 7810 per dimensioni, durevolezza, resistenza al piegamento, agli agenti chimici, ed altro[1].

Oltre alle usuali distinzioni dettate dalle diverse aziende produttrici, le tessere possono appartenere a due categorie fisiche, a seconda che posseggano o meno, oltre che alla memory card, anche un microprocessore, indispensabile nelle operazioni crittografiche e ottimizzato per svolgere questa funzione. Per cui le tessere dotate solo di scheda di memoria risultano essere un serbatoio di informazioni e vengono utilizzate appunto, per la registrazione sicura di dati; solitamente hanno una capienza fino a 4 KiloByte e grazie alla semplicità dei comandi e della struttura garantiscono, sebbene con prestazioni minori, un minore costo. Invece se dovesse essere necessario un maggiore livello di complessità delle operazioni, sarebbe opportuno utilizzare le Smart-Card dotate di microprocessore, che in forza di questo vengono utilizzate per

cucire funzionalità specifiche per necessità particolari. I comandi risultano quindi essere più complessi e coinvolgono anche l'utilizzo di chiavi, quali PIN e password, fino a comprendere l'uso di chiavi crittografiche e infrastrutture a chiave pubblica PKI (Public Key Infrastructure)[2].

Un ulteriore differenza è data dalla tipologia di trasmissione dati: Contact e Contactless. Le carte con protocolli Contact, il cui standard è definito in ISO/IEC 7816, hanno una placca d'oro da 1 cm quadrato che trasmette segnali elettrici per contatto e generalmente meno capienza in termini di memoria delle Contactless. Queste ultime sono invece caratteristiche per la trasmissione di dati sotto forma di onde elettromagnetiche, usualmente con frequenza a 13.56 MHz, secondo lo standard ISO/IEC 14443, ed ad una distanza massima di 10 cm; RFID, cioè Radio-frequency identification, descrive appunto l'utilizzo di campi elettromagnetici a frequenze radio, usati per trasportare dati. Smart-Card di questo tipo sono dotate di una piccola antenna, grazie alla quale quando si avvicinano ad un campo magnetico od elettromagnetico della giusta frequenza, viene trasferita energia alla carta, tramite induzione, che si accende ed immediatamente è inviato il segnale di clock e stabilito un protocollo di comunicazione fra il lettore e la carta, per lo scambio di dati e istruzioni codificate[3].

3. ISO/IEC 7816-4

Lo standard internazionale ISO/IEC 7816, creato dalla International Organization for Standardization (ISO) e dalla International Electrotechnical Commission (IEC), definisce le caratteristiche fisiche, le caratteristiche elettriche, le metodologie ed i protocolli, i comandi, e le tecniche ed operazioni di gestione delle Smart-card.

La parte quarta, nello specifico, tratta del contenuto dei messaggi scambiati fra lettore e carta: Command e Response, in termini di sequenze di byte, e della struttura e modalità di selezione dei file nella carta[4][5].

3.1. Struttura dati logica

3.1.1. Organizzazione FileSystem

Una struttura dati è un sistema per organizzare in maniera logica gruppi di byte; nel caso delle Smart-Card si utilizza il classico sistema gerarchico di directory annidate che contengono file semplici.

Possiamo distinguere quindi due tipologie di file semplici: DF ed EF, Dedicated File ed Elementary File. I primi sono le note “cartelle”, cioè file contenitore con al loro interno altri file e/o directory; particolare importanza ha la directory principale o root, detta MF: Master File, sempre presente, che contiene tutte le altre, e che, viene selezionata automaticamente al reset della carta: cioè non appena viene stabilito il contatto con il lettore.

Gli Elementary file sono, invece, gruppi di byte che rappresentano i veri e propri dati; possono essere classificati per scopo come Internal EF e Working EF, i primi sono file di dati utilizzati dalla carta per gestione e controllo, mentre i secondi sono file che immagazzinano dati utilizzati da applicazioni esterne. Per quanto riguarda la rappresentazione, le tessere supportano 4 tipologie di struttura per gli EF:

- Transparent Elementary File: semplice blocco di byte modificabile tramite offset
- Linear Fixed Elementary File: file diviso in blocchi di lunghezza fissa, con accesso per numero di blocco
- Linear Variable Elementary File: file diviso in blocchi di lunghezza variabile, con accesso per numero di blocco
- Cyclic Elementary File: file diviso in blocchi di lunghezza fissa, con accesso per numero di blocco, ad ogni accesso in lettura o scrittura si opera sul blocco successivo a quello su cui si ha lavorato all'accesso precedente

3.1.2. Metodi di referenza

Per poter operare su un determinato file in una determinata directory, è necessario avere a disposizione delle informazioni per raggiungerlo e leggere o scrivere byte in maniera chiara e precisa. Nelle Smart-Card questo è possibile in diverse modalità, a seconda delle necessità:

- file identifier (FID): il file può essere selezionato tramite un nome codificato in due byte, da 0x0001 a 0xFFFE; 0x0000 e 0xFFFF sono riservati per usi futuri (RFU), 0x3FFF è anch'esso riservato e 0x3F00 è il codice che identifica il Master file; con questa modalità si può selezionare qualsiasi EF o DF dentro la directory corrente, e per questo gli EF dentro una DF devono avere nomi diversi per garantire l'univocità della selezione
- path: la selezione per path, con argomento del comando di selezione costituito dalla concatenazione dei FID del percorso delle DF, dal Master file o dalla directory corrente al file da selezionare, offre una modalità di riferimento assoluta ed univoca, in cui l'ordine dei FID va dal genitore al figlio
- short EF identifier: valore codificato in 5 bit, da 1 a 30, in cui lo 0 rappresenta il file corrente; questa modalità non può essere usata in un path o come FID

- DF name: vero e proprio nome di un file, codificato da 1 a 16 byte, può essere usato come riferimento assoluto di un EF globalmente all'interno della carta
- application identifier (AID):metodo di selezione diretta delle DF che rappresentano applicazioni (ADF), secondo l'AID, che per tutte le applicazioni di una carta è registrato nel file 0x2F00, e permette di selezionare dal Master file ogni directory e da una directory solo quelle che essa contiene, secondo lo standard ISO/IEC 7816-5.

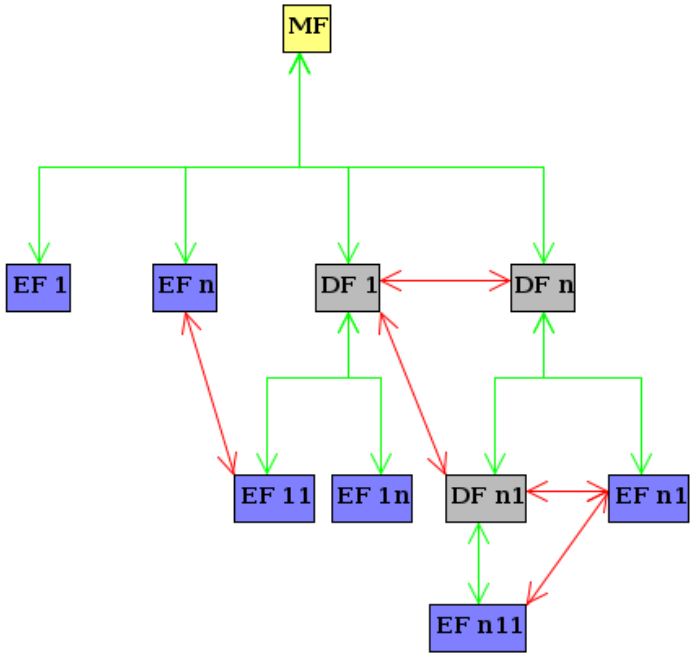

Figura 2: modalità di navigazione file system

3.1.3. File Control Parameter

Per la gestione specifica di un file deve essere necessario conoscere esattamente tutte le sue caratteristiche, per poterci lavorare in maniera sensata e precisa. Questo è garantito dalla presenza del File Control Parameter (FCP), in ogni elemento, file o directory, presente nella carta, che specifica gli attributi logici, strutturali e di sicurezza dell'elemento stesso[6]. Questi attributi prendono la forma di un template, costituito da una sequenza di byte ordinati per tipologia, che viene restituito come risposta a seguito di un comando di selezione di file.

Quindi per ogni attributo sono presenti: un byte che rappresenta il tag, cioè la categoria di attributo, un byte che rappresenta la lunghezza (in numero di byte) dell'attributo, e infine i byte che sono il valore dell'attributo stesso[4][6]; l'FCP è quindi costituito dalla concatenazione di tutti i byte di ogni attributo del file, secondo:

File Control Parameter X=obbligatorio, V=opzionale, N=non presente							
Tag			Lunghezza	Valore	EF	DF	ADF[1]
0x62			Variabile	Etichetta di inizio FCP	X	X	X
	0x80		02	Lunghezza del file, escluso FCP	X	N	N
	0x82		01	Categoria file	X	X	X
	0x83		02	FID	X	X	N
	0x84		05-10	AID	N	N	X
	0x88		00-01	Short EF identifier	V	N	N
	0x8A		01	Byte del ciclo di vita del file	V	V	N
	0xA1		Variabile	Permessi di accesso	X	X	X
		0x8C	Variabile	Interfaccia Contact	V	V	V
		0x9C	Variabile	Interfaccia Contactless	V	V	V

1 Dedicated file che rappresenta un applicazione

Byte di categoria file								
0	0							Non condivisibile Contact/Contactless
0	1							Condivisibile Contact/Contactless
1								Riservato per usi futuri
		0	0	0				Working EF
		0	0	1				Internal EF
		0	1	0				Proprietari
		0	1	1				
		1	0	0				
		1	0	1				EF
		1	1	0				
		1	1	1				DF
					0	0	0	Nessuna informazione
					0	0	1	Transparent file
					0	1	0	Linear fixed elementary file
					0	1	1	Linear fixed SIMPLE-TLV[1]
					1	0	0	Linear variable elementary file
					1	0	1	Linear variable SIMPLE-TLV
					1	1	0	Cyclic elementary file
					1	1	1	Cyclic SIMPLE-TLV

Particolare attenzione va invece data al byte del ciclo di vita di un file, EF o DF, che rappresenta lo stato operativo: attivato, deattivato e stato di terminazione [6]; questo byte non è richiesto obbligatoriamente alla creazione di un file (che in questo caso è automaticamente attivato) ma impone dei blocchi di accesso gestiti dai permessi, che vengono modificati quando è mutato lo stato operativo del file:

Byte del ciclo di vita di un file								
0	0	0	0	0	1	0	1	Stato operativo attivato
0	0	0	0	0	1	0	0	Stato operativo deattivato
0	0	0	0	1	1	0		Stato di terminazione

1 SIMPLE-TLV è la modalità di codifica: tag-lunghezza-valore per un attributo

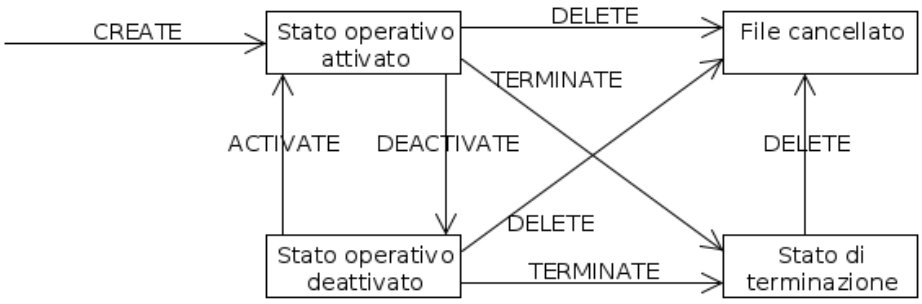

Figura 3: transizioni di stato di un file innescate da comandi APDU [6]

3.1.4. Attributi di sicurezza

Un particolare gruppo di attributi di un file è quello dei suoi attributi di sicurezza, con Tag A1, cioè dei permessi di accesso in lettura e scrittura[6]. Prima di spiegare le regole di interpretazione dei byte, è necessario introdurre alcuni concetti: lo Status di sicurezza è dato dalle possibilità e limitazioni di agire di un utente, può essere:

- globale, cioè valido per l'intera durata della sessione del canale di comunicazione con la carta
- per applicazione, quindi specifico per una certa funzione
- per file, limitato all'utilizzo di un certo file
- per comando, valido soltanto per un unico comando

Le flag dello status di sicurezza sono settate con il completamento con successo di una funzione di autenticazione; in questo documento viene data particolare attenzione al comando VERIFY[1], per la semplice autorizzazione all'utilizzo di un file tramite PIN, piuttosto che ai comandi EXTERNAL AUTHENTICATION, MUTUAL AUTHENTICATION ed INTERNAL AUTHENTICATION[2], che coinvolgono chiavi crittografiche e che sono al di là

1 Comando trattato in 3.3.1.
2 Comandi trattati in 3.3.7.

dello scopo di questo documento[6].

Il blocco di byte che comprende gli attributi di sicurezza è diviso in due parti rispettivamente per il protocollo Contact e Contactless, l'interpretazione dei byte per entrambi è la stessa; subito dopo il byte con la Tag A1 c'è un byte che rappresenta la lunghezza, intesa come numero di byte, in esadecimale degli elementi successivi che rappresentano gli attributi di sicurezza ed in seguito i due blocchi per Contact e Contactless.

Ciascun blocco è formato dal byte di Tag: 8C per Contact e 9C per Contactless, da un byte che rappresenta la lunghezza dei byte seguenti di permessi, ed infine da uno o più sottoblocchi di byte che codificano i diritti di accesso.

Questi sottoblocchi sono costruiti a partire da due particolari byte: AMB e SCB: rispettivamente Access Mode Byte e Security Condition Byte: il primo indica su quante e quali operazioni[1] sono presenti permessi, il secondo indica il tipo specifico di sicurezza per quella data operazione, in quanto in ogni sottoblocco è presente un AMB all'inizio, e tanti SCB quanti sono i bit settati a 1 su esso, e cioè i comandi, in maniera ordinata secondo:

Access Mode Byte per un DF								Access Mode Byte per un EF[2]							
0	1						DELETE	0							DELETE
		1					TERMINATE		1						TERMINATE
			1				ACTIVATE			1					ACTIVATE
				1			DEACTIVATE				1				DEACTIVATE
					0		Riservato per usi futuri					0			Riservato per usi futuri
						1	CREATE EF						1		UPDATE BINARY
							0 Riservato per usi futuri							1	READ BINARY

1 Comandi trattati in 3.3.

2 AMB per oggetti crittografici non sono stati trattati

Un SCB è invece un contenitore di informazioni sulle modalità di accesso ad un file, per una determinata funzione, secondo:

Security Condition Byte: condizioni di utilizzo di un comando								
0	0	0	0	0	0	0	0	Libero utilizzo del comando
1	1	1	1	1	1	1	1	Utilizzo del comando vietato
				0	0	0	0	Combinazione vietata
				diseguali				Riservato per uso proprietario
				1	1	1	1	Riservato per usi futuri
	1							Utilizzo del comando previa autenticazione device[1]
		1						Utilizzo previa EXTERNAL AUTHENTICATION[2]
			1					Utilizzo previa Autenticazione utente[3]
0								Almeno una condizione[4]
1								Tutte le condizioni

1 Autenticazione acquisita da un device esterno, non utile alla trattazione
2 Comando trattato in 3.3.7.
3 Autenticazione tramite PIN o Password
4 Fra le tre precedenti

3.2. Il protocollo logico

Per un corretto dialogo fra carta e lettore è stato definito un protocollo di comunicazione, poi reso universale nello standard 7816-3 [7], che prevede svariate norme per la precisa e sicura comunicazione fra i due device.

Prima di entrare nel dettaglio è necessario dare alcune utili definizioni: il Card Acceptance Device (CAD)[1], cioè il lettore di schede, è il device che accende la carta e le dà energia per lo scambio, in un modello half-duplex e nel classico schema master-slave, in cui il CAD, che è il master, invia richieste (comandi) e la carta è lo slave che riceve le richieste ed invia le risposte[8].

I messaggi inviati sia dal CAD che dalla tessera sono detti APDU: Application Protocol Data Unit, e lo scambio reciproco rientra in quattro casistiche, a seconda che vengano trasmessi o meno dati sensibili, oltre che ai byte di controllo:

Casistiche APDU Command e Response		
1	CAD non invia dati	Smart-card non invia dati
2	CAD non invia dati	Smart-card invia dati
3	CAD invia dati	Smart-card non invia dati
4	CAD invia dati	Smart-card invia dati

La tipologia di trasmissione dati e le modalità sono incluse nel TPDU: Transmission Protocol Data Unit, che regola la trasmissione degli APDU, ed è definito nello standard 7816-3 [7]; i protocolli maggiormente usati sono T=0, orientato ai byte (o caratteri) e che ha bisogno di scambio di messaggi extra per un trasferimento più massiccio di dati, e T=1, più nuovo, orientato a blocchi, e che quindi supporta dimensioni maggiori di dati.

Il TPDU serve quindi alla carta ed al lettore per accordarsi sulle modalità di scambio ed è presente di norma nello ATR: Answer To Reset, messaggio di 33 byte inviato dalla Smart-Card non appena viene accesa dalla vicinanza con un lettore, che contiene inoltre la frequenza di trasmissione e dati hardware della carta [8].

1 Trattato in 4.1.1.

3.2.1. Apdu Command e Response

Tutti i messaggi APDU, Command o Response, sono strutturati secondo lo standard descritto in ISO 7816-4, che opera una divisione del messaggio in intestazione(header) e corpo(body), per il Command, e corpo(body) e coda(trailer), per la Response[4];

Per il comando, l'intestazione comprende i byte che lo caratterizzano e ne specificano i parametri, mentre il corpo è il contenuto effettivo dei dati da scambiare. Invece per la risposta, il corpo contiene i dati e la coda è costituita dai byte di stato[1], che codificano lo specifico risultato per quel determinato comando.

Il comando è costituito in particolare da:

Header	CLA	1 byte	Classe del comando
	INS	1 byte	Tipo di istruzione
	P1	1 byte	Primo parametro
	P2	1 byte	Secondo parametro
Body	Lc	1/3 byte	Lunghezza del campo data
	DATA	Lc byte	Campo che contiene i dati da trasmettere
	Le	1/3 byte	Lunghezza attesa

il campo CLA codifica la classe del comando e contiene informazioni riguardo al tipo di sicurezza, al canale logico[2], e specifica se e quanto il comando è coerente con lo standard[6]:

0	0	0	0					Il comando è l'ultimo di una concatenazione
			1					Il comando non è l'ultimo di una concatenazione
				0	0			Nessun tipo di sicurezza
				0	1			Formato sicurezza proprietario
				1	0			Comando non autenticato
				1	1			Comando autenticato
						x	x	Numero canale logico

1 Trattato in 3.2.2

2 Se supportato dalla carta, permette l'esecuzione concorrente di più applicazioni

il campo INS codifica il comando da eseguire[4][1]:

VERIFY	0x20
READ BINARY	0xB0
CREATE FILE	0xE0
UPDATE BINARY	0xD6
ERASE BINARY	0x0E
SELECT FILE	0xA4

i campi P1 e P2 costituiscono due byte di opzioni aggiuntive della specifica funzione[2].

Per quanto riguarda invece la risposta:

Body	DATA	Lr byte	Campo con i dati in risposta
Trailer	SW1	1 byte	Stato della risposta
	SW2	1 byte	Specifiche della risposta

Il campo dati contiene i byte di informazioni richieste dal comando precedente, mentre le Status Word (o Status Byte), caratterizzano il tipo di risposta, in SW1, e le specifiche della effettività o meno del comando di cui compongono la risposta, in SW2.

1 Per semplicità sono riportati solo i comandi rilevanti per la trattazione
2 Le opzioni di ciascun comando sono trattate specificatamente in 3.3.1-7

Alla luce di questi dati si può specificare ed aggiornare la classificazione fatta nel paragrafo 3.2 riguardo la coppia Command/Response in base al contenuto o meno di dati, nel messaggio APDU:

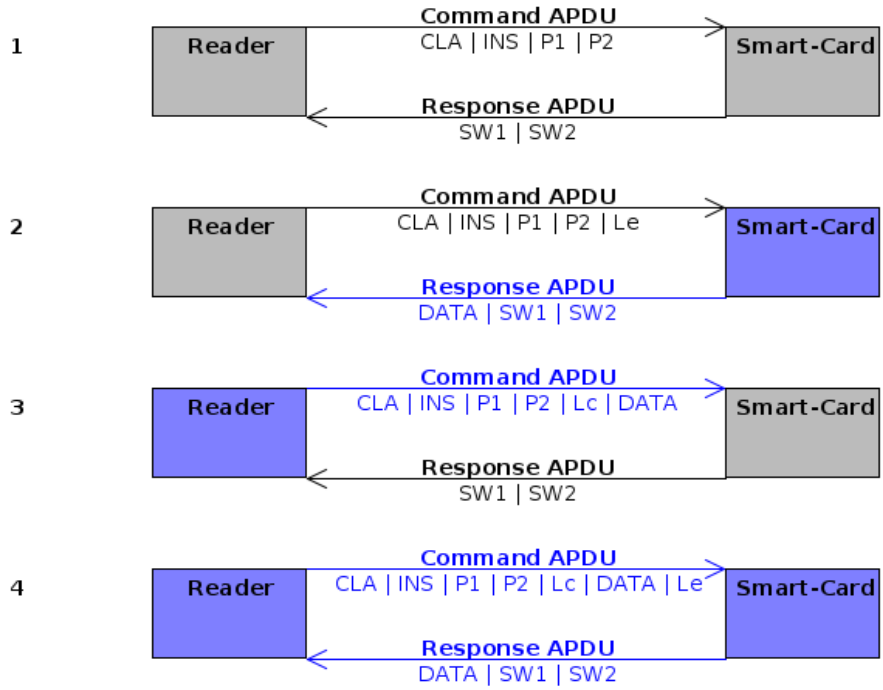

Figura 4: categorie APDU in base al trasporto di dati [8]

3.2.2. La Status Word

Per un corretto utilizzo degli strumenti sopra citati, è necessario essere a conoscenza anche della codifica delle Status Word o Status Byte, accodate al campo data in ogni messaggio APDU di risposta; questi byte indicano, oltre che il successo o meno di una operazione, anche l'eventuale cambiamento di stato della carta, e sono da intendere come SW1-SW2 come segue [4]:

Elaborazione normale	0x9000	Nessun requisito ulteriore richiesto
	0x61XX	SW2 indica il numero di byte in risposta ancora disponibili

Elaborazione con avvertimento	0x62XX		Memoria non volatile non modificata
		0x00	Nessuna informazione
		0x81	Dati nella risposta possono essere corrotti
		0x82	Raggiunta EoF[1] prima di leggere Le byte
		0x83	File selezionato è stato invalidato
		0x84	FCI[2] formattato in maniera non corretta
	0x63XX		Memoria non volatile modificata
		0x00	Nessuna informazione
		0x81	File riempito dall'ultima scrittura
		0xCX	Contatore a X[3]

Errori in esecuzione	0x6400		Memoria non volatile non modificata
	0x65XX		Memoria non volatile modificata
		0x00	Nessuna informazione
		0x81	Errore in memoria
	0x66XX		Riservato per problemi di sicurezza

1 End of File, fine del file o del record
2 File Control Information: generico per FCP
3 Interpretazione di X dipende dal comando

Errori di controllo	0x6700		Lunghezza errata
	0x68XX		Funzioni della Classe non supportate
		0x00	Nessuna informazione
		0x81	Canale logico non supportato
		0x82	Messaggi in sicurezza non supportati
	0x69XX		Comando vietato
		0x00	Nessuna informazione
		0x81	Comando non compatibile con struttura del file
		0x82	Status di sicurezza non valido
		0x83	Metodo di autenticazione bloccato
		0x84	Dati sono stati invalidati
		0x85	Condizioni d'uso non valide
		0x86	Comando non valido (nessuna EF)
		0x87	Dati mancanti su oggetti SM[1]
		0x88	Dati incorretti su oggetti SM
	0x6AXX		Parametri P1, P2 errati
		0x00	Nessuna informazione
		0x80	Parametri incorretti nel campo dati
		0x81	Funzione non supportata
		0x82	File non trovato
		0x83	Record non trovato
		0x84	Spazio di memoria nel file non sufficiente
		0x85	Lc inconsistente con TLV[2]
		0x86	Parametri P1, P2 incorretti
		0x87	Lc inconsistente con P1, P2
		0x88	Dati riferiti non trovati
	0x6B00		Parametri P1, P2 errati
	0x6CXX		Lunghezza attesa errata, SW2 indica la corretta lunghezza
	0x6D00		Campo INS non valido
	0x6E00		Classe non supportata
	0x6F00		Nessuna diagnosi precisa

1 Secure messaging object: file coinvolti con la sicurezza, chiavi crittografiche, pin, e altro
2 Modalità di codifica: tag-lunghezza-valore di un attributo

3.3. Comandi utilizzati

Il fulcro della logica che supporta le Smart-Card, è caratterizzato dai comandi APDU, specificati nello standard ISO/IEC 7816-4 sez. 6; è importante notare che non tutti i comandi descritti nello standard devono essere necessariamente supportati e implementati nelle specifiche tessere, in quanto a differenti necessità vengono affiancati differenti strumenti; a seconda della tipologia di comando ed in base all'evento ed all'effetto che produce sulla carta, possiamo classificarli come:

- comandi di autenticazione utente, come VERIFY, che coinvolgono il PIN utente, ed il suo contatore di immissioni errate[1]
- comandi di autenticazione, che gestiscono le chiavi crittografiche e la firma digitale in generale
- comandi relativi ad operazioni di sicurezza: cifratura e algoritmi di hash
- comandi per la gestione di file e directory, per selezionare, creare, scrivere e leggere file all'interno della carta, come SELECT, READ BINARY, WRITE BINARY, UPDATE BINARY ed ERASE BINARY
- comandi per leggere, scrivere e generare particolari file, come chiavi crittografiche

Di seguito sono riportati nei particolari i principali comandi utili alla trattazione.

1 Trattato in 3.3.1.

3.3.1. VERIFY

Il comando VERIFY confronta il campo data che fa parte del suo corpo con il file di sicurezza specifico; così facendo permette all'utente di autenticarsi alla carta ed acquisire specifici permessi di accesso ed esecuzione. In caso di autenticazione andata a buon fine, lo status di sicurezza viene modificato, mentre in caso negativo non solo l'utente non è ovviamente autenticato, ma viene decrementato il contatore di tentativi di immissione, che vale inizialmente 3, e che arrivato a 0 non permette più all'utente di autenticarsi. Lo si può in seguito ripristinare con i comandi RESET RETRY COUNTER e CHANGE REFERENCE DATA[1].

Nello specifico, i byte che compongono il comando sono codificati come segue[4]:

Header	CLA	Descritto in 3.2.1.
	INS	0x20
	P1	0x00
	P2	Quantificatore del campo DATA
Body	Lc	Lunghezza del campo DATA
	DATA	Informazione di sicurezza[2]
	Le	Vuoto

Per il secondo parametro, la codifica è:

0	0	0	0	0	0	0	0	Nessuna informazione
0								Riferimento globale[3]
1								Riferimento specifico (DF specifica)
			X	X	X	X	X	Numero del riferimento[4]
Altri valori								RFU

1 Trattati in 3.3.7.
2 Tipicamente Pin o Password
3 Status di sicurezza globale trattato in 3.1.4.
4 Identifica lo specifico PIN o Password

Nel caso in cui il parametro P2 sia 0x00, si intende indicare che l'informazione di sicurezza è univoca e non sono quindi necessari ulteriori specifiche.

Per quanto riguarda la risposta, essa è codificata come da norma dal campo dati, vuoto, e dai due byte di stato SW1-SW2 come segue:

0x90	0x00	Autenticazione effettuata
0x63	0x00	Autenticazione fallita
	0xCX	Autenticazione fallita, contatore settato a X
0x69	0x83	Autenticazione bloccata
	0x84	Dati riferiti[1] non validi
0x6A	0x86	Parametri P1-P2 incorretti
	0x88	Dati riferiti non trovati

Oltre che per la funzione di autenticazione, il comando VERIFY, con il corpo vuoto, può essere utile anche per acquisire dei dati, tramite i byte di stato: se in risposta si ottiene 0x9000, indica che l'autenticazione è già stata effettuata, mentre se si riceve 0x63CX, si ricava il valore del contatore di tentativi di immissione senza modificarlo.

1 Specifico PIN o Password con cui si vuole confrontare quello immesso

3.3.2. Read Binary

La risposta al comando READ BINARY eseguito con successo, contiene i byte del file letto se è di tipo Transparent Elementary File[1], in caso contrario la lettura fallisce; il comando può inoltre essere nullo se lo status corrente di sicurezza non soddisfa l'attributo di sicurezza per la funzione di lettura di quel file. Per poter leggere un file, lo si deve prima, ovviamente, selezionare[2],oppure si può specificare uno Short EF Identifier[3] nei byte parametro P1-P2; inoltre, essendo le caratteristiche della carta non illimitate, è necessario specificare l'offset di partenza della lettura e la dimensione del blocco da leggere, entrambi quantificati in byte. La struttura del comando è fedele a[4]:

Header	CLA	Descritto in 3.2.1.
	INS	0xB0
	P1	Parametro P1
	P2	Parametro P2
Body	Lc	Vuoto
	DATA	Vuoto
	Le	Numero di byte da leggere

Considerando i byte parametro come segue:

Parametro P1				Parametro P2
0	offset del file precedentemente selezionato			
1	0	0	1 < Short File Identifier < 30	Offset del file selezionato

1 Trattato in 3.1.1.
2 Comando SELECT trattato in 3.3.6.
3 Trattato in 3.1.2

La risposta sarà invece composta dal campo data con gli Le byte letti, e gli usali due byte di stato SW1 e SW2, codificati secondo:

Tipologia	SW1	SW2	Descrizione
Normale esecuzione	0x90	0x00	Lettura effettuata
Warning	0x62	0x81	Parte dei dati in risposta potrebbero essere corrotti
		0x82	Raggiunta EoF[1] prima di leggere Le byte
Errori	0x67	0x00	Lunghezza nel campo Le errata
	0x69	0x81	Comando incompatibile con la struttura del file
		0x82	Status di sicurezza non valido
		0x86	Comando vietato (nessun EF corrente)
	0x6A	0x81	Funzione non supportata
		0x82	File non trovato
	0x6B	0x00	Parametri errati (offset maggiore della lunghezza del file)
	0x6C	0xXX	Lunghezza Le errata, XX rappresenta la corretta lunghezza

1 End of file

3.3.3. Create file

Per la creazione di file, sia EF che DF, viene utilizzato il comando Create file [4], che permette di generare un elemento, previa selezione di un DF, solamente se lo status di sicurezza soddisfa l'attributo di sicurezza Create EF[1], della directory corrente. In caso di successo, quindi, nella DF selezionata in precedenza, viene creato il nuovo EF o DF, che diventa il file corrente. Il comando è così strutturato[4]:

Header	CLA	Descritto in 3.2.1.
	INS	0xE0
	P1	0x00
	P2	0x00
Body	Lc	Lunghezza del campo data
	DATA	FCP[2] del file da creare
	Le	Vuoto

La risposta sarà sempre costituita dal campo dati, vuoto, e da i due byte di stato secondo:

Tipologia	SW1	SW2	Descrizione
Normale esecuzione	0x90	0x00	Creazione effettuata
Warning	0x69	0x82	Status di sicurezza non valido
	0x69	0x85	Condizioni d'uso non valide
	0x6A	0x89	File già esistente[3]
	0x6A	0x84	Memoria piena
	0x6A	0x86	Parametri P1-P2 incorretti
	0x6A	0x80	FCP errato

1 Trattato in 3.1.4.
2 File Control Parameter, trattato in 3.1.3.
3 Stesso identificativo FID o short FID

3.3.4. Update Binary

Per aggiornare e quindi modificare il contenuto di un file, si può ricorrere al comando Update Binary, che come il comando Read Binary, agisce su elementi di tipo Transparent File, solo se lo status di sicurezza è conforme all'attributo di sicurezza Update, per quel file. Sempre come per il comando di lettura, il file deve essere selezionato prima o specificato con un identificatore nei parametri del comando, ed in questo caso, l'EF letto diventa il file corrente. Le specifiche sono strutturate come segue[6]:

Header	CLA	Descritto in 3.2.1.
	INS	0xD6
	P1	Parametro P1
	P2	Parametro P2
Body	Lc	Lunghezza del campo data
	DATA	Sequenza di byte da scrivere
	Le	Vuoto

Come per il comando di lettura, a seconda che il file obiettivo sia stato o meno selezionato in precedenza, e quindi considerando la presenza di un identificativo di file direttamente nel messaggio APDU, nel secondo caso, i due byte di parametri vanno intesi come:

Parametro P1				Parametro P2
0	offset del file precedentemente selezionato			
1	0	0	1 < Short File Identifier < 30	Offset del file selezionato

Per quanto riguarda la risposta, possiamo notare l'usuale pattern di campo dati vuoto e status byte SW1-SW2 codificati come segue:

Tipologia	SW1	SW2	Descrizione
Normale esecuzione	0x90	0x00	Scrittura effettuata
Warning	0x62	0xCX	X è un contatore di tentativi di scrittura
Errori	0x65	0x81	Errore in memoria
	0x67	0x00	Campo Le errato
	0x69	0x81	Comando non compatibile con la struttura del file
		0x82	Status di sicurezza non verificato
		0x86	Comando non permesso
	0x6A	0x81	Funzione non supportata
		0x82	File non trovato
	0x6B	0x00	Offset eccede la lunghezza del file

3.3.5. Delete File

Come per gli altri comandi, anche per quello di cancellazione è necessario che gli attributi del file e lo status di sicurezza siano concordi; questo comando elimina l'ultimo file selezionato, EF o DF, e tutti i file al suo interno, nel caso di una directory; è importante notare che dopo l'esecuzione non viene settato nessun EF come file corrente ma la directory che conteneva l'elemento cancellato. La struttura del comando APDU è la seguente:

Header	CLA	Descritto in 3.2.1.
	INS	0xE4
	P1	0x00
	P2	0x00
Body	Lc	0x00
	DATA	Vuoto
	Le	Vuoto

Mentre la risposta APDU è costituita dal campo dati, vuoto, e dagli usuali byte di stato:

Tipologia	SW1	SW2	Descrizione
Normale esecuzione	0x90	0x00	Cancellazione effettuata
Errori	0x69	0x82	Status di sicurezza non valido
		0x85	Condizioni d'uso non valide
	0x6A	0x86	Parametri P1-P2 incorretti
	0x67	0x00	Lunghezza errata

3.3.6. Select File

La selezione di un file, tramite i metodi di referenza sopra descritti, setta come file corrente il file specificato nel comando, in modo tale che tutti i comandi successivi, abbiano come riferimento quell'elemento, fino ad una prossima eventuale selezione. Per quanto riguarda il cambiamento dello status di sicurezza, a seguito di una selezione di EF, esso viene mantenuto; invece, per le DF, viene mantenuto solo se la directory corrente attuale è contenuta o identica alla vecchia directory corrente, mentre si considera lo status della nuova negli altri casi. In generale, per una DF, è mantenuto lo status di sicurezza comune fra le directory che contengono la precedente e la nuova. La codifica per il comando di selezione è la seguente[4]:

Header	CLA	Descritto in 3.2.1.
	INS	0xA4
	P1	Parametro P1
	P2	Parametro P2
Body	Lc	Vuoto o lunghezza del campo data
	DATA	FID o path dal Master File o path dal DF corrente o DF name
	Le	Vuoto o massima lunghezza in risposta

Il campo data, come anche i campi Lc ed Le sono codificati tenendo in considerazione le scelte del parametro P1 descritto di seguito:

0	0	0	0	0	0	X	X	Selezione per FID
						0	0	Selezione MF, EF o DF
						0	1	Selezione di un DF contenuto in quello corrente
						1	0	Selezione di un EF contenuto in quello corrente
						1	1	Selezione del DF che contiene l'attuale DF (campo dati vuoto)
0	0	0	0	0	1	X	X	Selezione per DF name
					1	0	0	Selezione diretta per nome
0	0	0	0	1	X	X	X	Selezione per path
				1	0	0	0	Selezione per percorso dal master file
				1	0	0	1	Selezione per percorso dal DF corrente
Altri valori								Riservato per usi futuri

Risulta importante notare che il FID che si richiede di selezionare deve essere unico fra i nomi delle DF contenute nella directory corrente e che sono contenute in quella che contiene quest'ultima; inoltre se i byte parametro sono codificati come 0x0000 e il campo dati è vuoto, viene selezionato automaticamente il Master File. Per quanto riguarda il parametro P2, possiamo dire che è necessario per la selezione dei file divisi per record e per la scelta del template nel campo dati dell'APDU di risposta[4]:

0	0	0	0			0	0	Primo record
0	0	0	0			0	1	Ultimo record
0	0	0	0			1	0	Record successivo
0	0	0	0			1	1	Record precedente
0	0	0	0	X	X			Opzione delle informazioni di controllo in risposta
				0	0			Template FCI[1] in risposta
				0	1			Template FCP in risposta
				1	0			Template FMD[2] in risposta
Altri valori								Riservato per usi futuri

Il messaggio APDU di risposta è invece composto dall'usuale campo dati, stavolta con le informazioni dettate dal parametro P2, seguito da i byte di stato intesi come:

Tipologia	SW1	SW2	Descrizione
Normale esecuzione	0x90	0x00	Selezione effettuata
Warning	0x62	0x83	Il file selezionato è stato invalidato
		0x84	Campo informazioni di controllo non corretto
Errori	0x6A	0x81	Funzione non supportata
		0x82	File non trovato
		0x86	Parametri P1-P2 non corretti
		0x87	Lunghezza Lc inconsistente con i parametri P1-P2

1 Diverso tipo di informazioni di controllo
2 Diverso tipo di informazioni di controllo

3.3.7. Altri comandi

Lo standard ISO/IEC 7816-4 comprende, oltre che varie caratteristiche e metodologie per l'utilizzo delle Smart-Card, anche il set dei comandi standard descritto nelle pagine precedenti, tuttavia non è obbligatorio per tutti i produttori di tessere, far si che i loro prodotti supportino nel dettaglio i comandi descritti, quindi esistono diverse convenzioni specifiche al variare delle aziende coinvolte.

In questa trattazione sono stati selezionati i comandi più inerenti che sono poi stati usati per costruire l'applicativo descritto nella seconda parte di questo documento, tralasciando ulteriori comandi sia dello standard internazionale[4] che della convenzione europea fra produttori[6]. I comandi trattati sin ora sono stati descritti seguendo le indicazioni dello standard internazionale, fin dove possibile, applicando la convenzione europea qualora ci fosse bisogno di una maggiore precisione di descrizione in vista della parte di questa trattazione relativa alla produzione dell'applicativo.

È doveroso quindi far notare l'esistenza di molteplici comandi di uso generale presenti nello standard internazionale, ad esempio per la gestione dei record, e di comandi specifici per la gestione delle carte, prodotte da specifiche aziende, nella convenzione europea, come il comando per resettare il contatore di tentativi di immissione di Password della carta.

Alla luce di ciò, viene tralasciata la descrizione di quei comandi e quelle caratteristiche operative, sia internazionali che specifiche, che sarebbero fuorvianti rispetto all'obiettivo di questo documento che resta quello di documentare la produzione dell'applicativo discussa di seguito.

4. File Manager

L'obiettivo principale di questa trattazione è quello di documentare la produzione di un applicativo per leggere e scrivere dati sulle Smart-Card. Fin ora sono stati discussi gli aspetti teorici e logici, e le regole di utilizzo e di codifica che stanno prima del vero e proprio sviluppo del prodotto finale; con la teoria acquisita si può quindi scendere nei particolari del progetto, dopo aver appuntato alcune necessarie note.

4.1. Materiale

Fondamentale allo sviluppo del prodotto, sono le basi hardware e software, che ne costituiscono l'architettura e comprendono il lettore e la carta da un lato, e il linguaggio, l'ambiente e i driver dall'altro.

4.1.1. Hardware

Strumento essenziale per questo tipo di progetto è lo Smart-Card reader, o Card Acceptance Device (CAD), che nonostante la nomenclatura “reader”, opera sia in lettura che scrittura; in particolare è stato scelto il lettore HID Omnikey® 5321 [9] con interfaccia duale Contact/Contactless e cavo USB per la connessione ad un calcolatore che legge e scrive a 13.56 MHz, in linea con lo standard internazionale.

Figura 5: Reader HID Omnikey® 5321

Accoppiato al lettore, c'è l'ormai nota Smart-Card o Integrated Circuit Card (ICC), tessera che rappresenta il nucleo della trattazione. Risulta doveroso far notare che scrivere un programma con interfaccia user-friendly per lettura e scrittura di dati su tessere, di ogni tipo, sarebbe notevolmente difficile, se non impossibile, perché nonostante la maggioranza di produttori si attenga allo standard, non sono rari i casi in cui i comandi APDU specifici per una carta siano diversi da quelli internazionali, ed in più alcune aziende sviluppano schede con comandi aggiuntivi o senza supporto a certi comandi noti. A fronte di questo si è deciso di scegliere come caso di studio la Enjoy My Unicam Card[10], carta servizi universitaria nata dalla convenzione tra Unicam: Università degli studi di Camerino, UBI: Banca Popolare di Ancona, e Namirial® s.p.a.: società informatica di produzione software[11].

La tessera è una carta prepagata multifunzione che associa alle funzioni riguardanti i servizi universitari[1], le normali possibilità di una tessera bancaria, cioè pagamenti, bonifici e transazioni in generale; per quanto riguarda questo progetto viene utilizzata come contenitore di dati di ogni sorta, sfruttando lo spazio residuo in memoria.

Figura 6: Carta Enjoy My Unicam

1 Si intende servizio mensa, tasse, documenti ed altro

4.1.2. Software

Per un effettivo funzionamento dell'ambiente Card/Reader sono ovviamente necessari:

- ambiente di sviluppo Java[1], per la programmazione di base
- driver del lettore HID Omnikey® 5321 [9]
- Java Smart Card I/O API, libreria per la comunicazione fra applicazioni Java e lettori di Smart-Card[12]
- librerie per la gestione della grafica in Java: Swing[14] e Awt[15]

1 Linguaggio scelto per favorire la portabilità

4.2. Programmi utili

Per una analisi ed una fase di test preventiva alla produzione dell'applicativo, sono stati sicuramente necessari alcuni software aggiuntivi, che sebbene non siano stati incorporati nel prodotto finale, hanno avuto un ruolo fondamentale nella parte di modellazione e comprensione dei comandi necessari e dell'ambiente in generale.

Sicuramente uno dei più importanti software è pcsc-tools, che comprende una raccolta di strumenti di analisi e diagnostica per ambienti PC/SC[1][13], fra i quali:

- pcsc_scan, che scansiona tutti i lettori connessi alla macchina sulla quale viene lanciato, non appena viene inserita una tessera, e riporta tutte le caratteristiche tecniche, logiche e di utilizzo di quest'ultima
- ATR_analysis, per scorrere l'ATR di una carta[2]
- smartcard_list.txt, non un tool ma un elenco di ATR noti
- scriptor, interfaccia a riga di comando per inviare messaggi APDU alla carta e leggere le risposte
- gscriptor, interfaccia grafica Perl-Gtk2 per scriptor

Un altro tool molto utile è JSmartCardExplorer[16], programma a basso livello scritto in Java, che tramite una interfaccia grafica, consente di inviare e ricevere messaggi APDU codificati in esadecimale byte a byte; il particolare punto di forza di questo tool sta nel fatto che non solo dà la possibilità di conservare i comandi eseguiti, ma permette anche di tradurre gli esadecimali del campo dati dei messaggi nello standard ASCII, sia in lettura che in esecuzione.

1 Personal Computer/Smart Card
2 Trattato in 3.2.

4.3. GUI

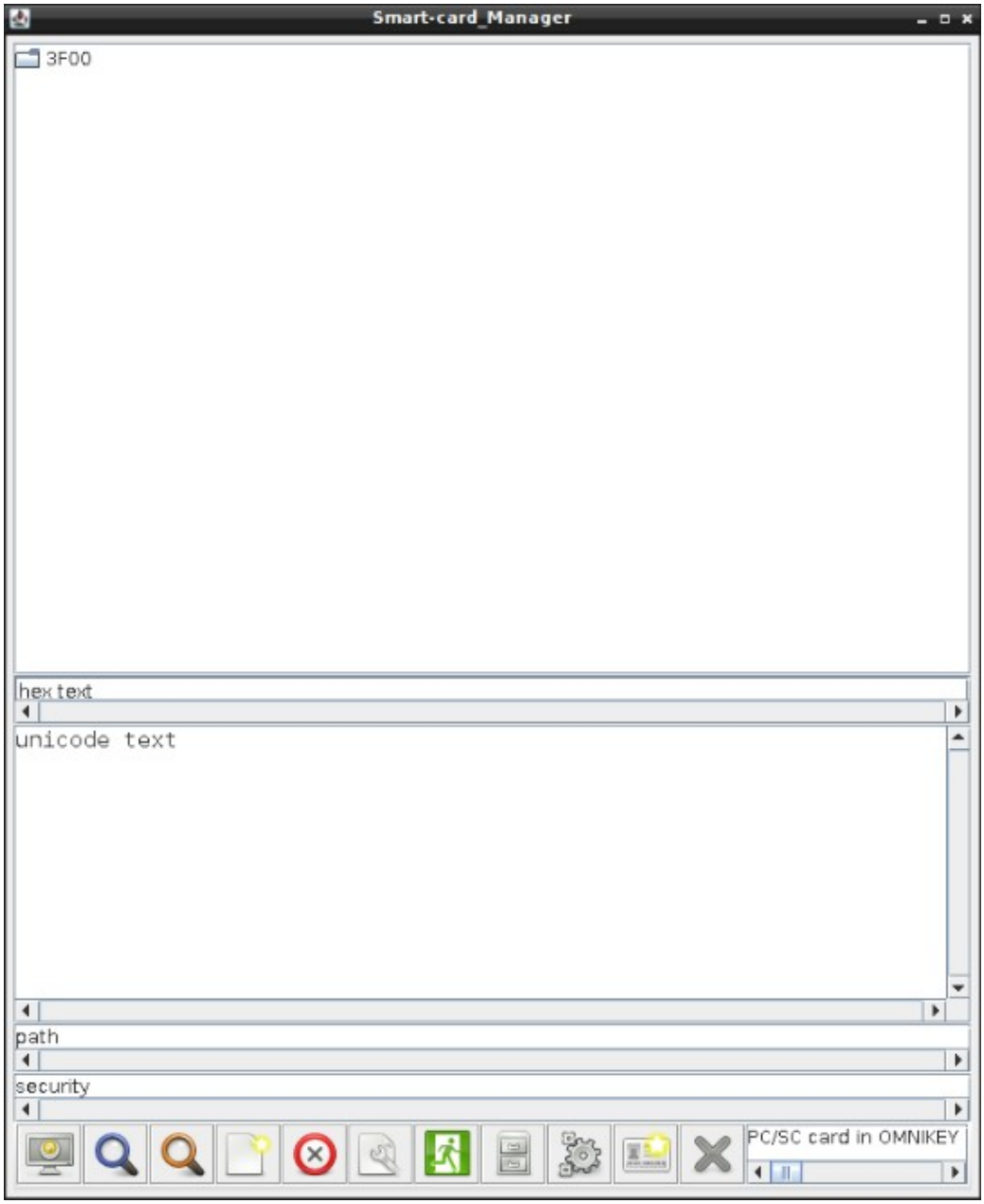

Figura 7: interfaccia principale

Dopo una prima introduzione alla logica che sta dietro ai messaggi APDU, limitata alle definizioni utili, possiamo ora presentare l'applicativo creato, scritto in Java, per garantire portabilità, e appoggiato alle classiche librerie grafiche Swing[14] e Awt[15], per la costruzione della statica e dinamica delle finestre e popup.

L'interfaccia principale si presenta come una finestra standard con i seguenti campi:

- un area che ospiterà l'albero delle directory, subito dopo aver lanciato uno dei comandi di scansione che presentano il FileSystem
- un area di testo che offre il contenuto di un file selezionato in formato esadecimale, e viene aggiornata ogni volta che si seleziona un file o directory dall'albero
- un area di testo che traduce automaticamente, alla selezione del file dall'albero, il suo contenuto nello standard Unicode
- un area di testo che contiene il percorso del file selezionato dalla directory principale
- un area di testo con gli attributi di sicurezza[1] del file selezionato
- una barra con un' area di notifica e tutti i bottoni delle varie funzioni del programma

Quando viene selezionato un file dall'albero, vengono aggiornate tutte le aree di testo: il contenuto, se è un EF, viene inserito nella sua specifica area, sia in esadecimale che in Unicode, il percorso viene aggiornato e gli attributi di sicurezza vengono presentati; in oltre viene scritto nell'area di notifica un messaggio che riporta il successo della lettura, in caso positivo o il problema riscontrato, in caso negativo. Le aree con i contenuti esadecimale e Unicode sono sincronizzate, per la scrittura, in modo da conoscere la stringa di byte che si andrà a scrivere, digitando solo caratteri nell' area Unicode; un ruolo particolare è stato invece assegnato all'area di notifica, che presenta inizialmente informazioni sul lettore e sulla carta che ha riconosciuto, ed in seguito offre notizie sulla riuscita o meno delle varie operazioni, riportando anche gli specifici problemi riscontrati nell'esecuzione.

La barra dei bottoni contiene i lanciatori per tutte le funzioni implementate dall'applicativo e corrispondenti ad altrettanti messaggi APDU, tranne il comando di lettura[2], che viene innescato alla selezione[3].

1 Trattato in 3.1.4.
2 Trattato in 3.3.2.
3 Trattato in 3.3.6.

La barra è quindi composta dai seguenti lanciatori:

- Authenticate: 

  bottone che permette l'autenticazione utente tramite PIN[1], visualizzando un prompt per l'immissione del codice

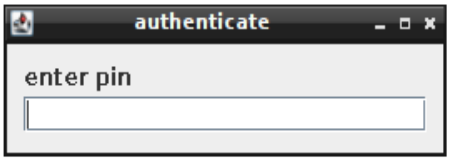

Figura 8: prompt per l'immissione del PIN

- Scan: 

  lancia la funzione di scansione della carta[2]

- FastScan: 

  lancia la funzione di scansione veloce[3]

- Create: 

  visualizza un prompt per la creazione di EF o DF[4], che chiede di immettere le specifiche dell'elemento da creare

Figura 9: prompt per la creazione

- Destroy: 

  invoca il comando per eliminare il file selezionato[5]

- Modify: 

  comando di scrittura di un file con il testo nella specifica area, modificato in precedenza[6]

1 Trattato in 3.3.1.
2 Trattato in 4.4.2.
3 Trattato in 4.4.2.
4 Trattato in 3.3.3.
5 Trattato in 3.3.5.
6 Trattato in 3.3.4.

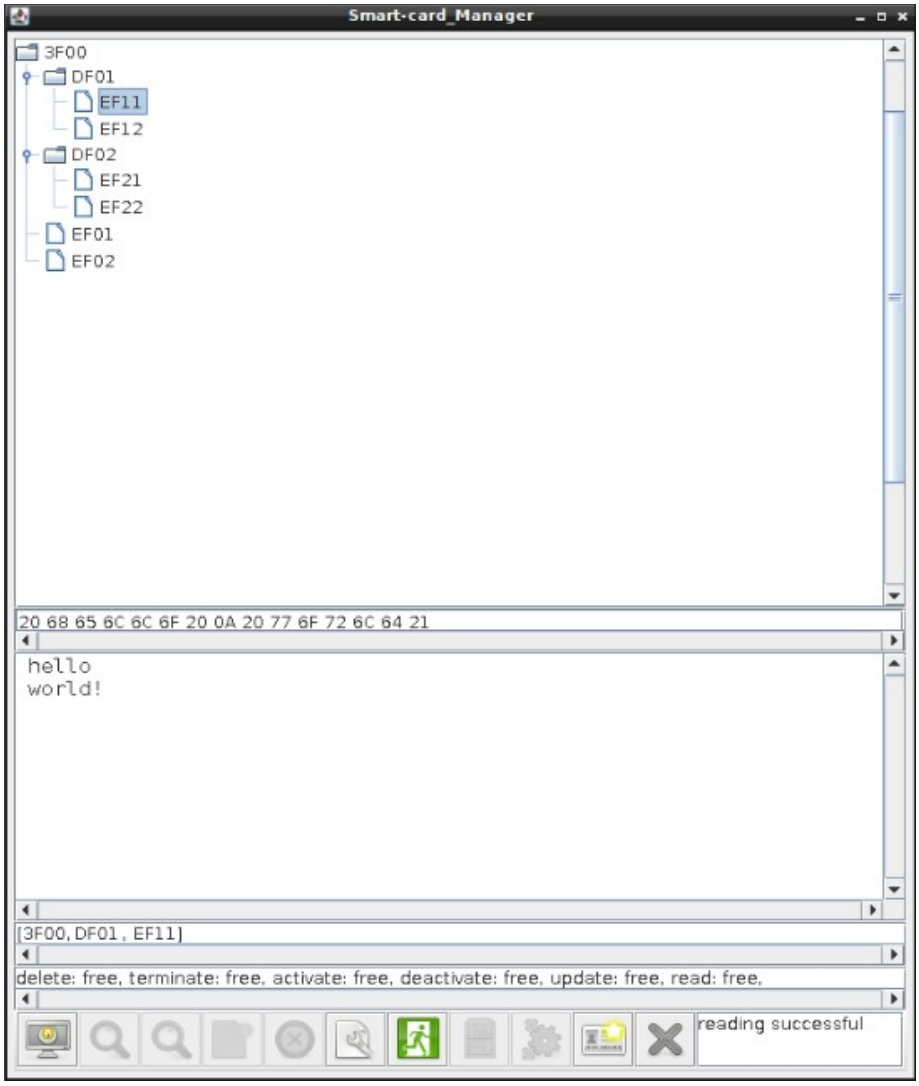

Figura 10: lettura del file EF11 appena modificato

Tutti i comandi trattati in precedenza sono funzioni che lavorano sulla carta solo nella sessione attuale e non hanno ripercussioni sul futuro utilizzo del software applicativo; i comandi che verranno ora analizzati sono invece basati sul concetto di FileSystem, relativo alle Smart-Card, che verrà discusso in seguito[1], ed impongono l'autenticazione utente tramite prompt per ogni operazione:

- Write FileSystem:

  questo lanciatore invoca una funzione che scansiona[2] la carta e crea l'albero delle directory; inoltre viene creato un record nella tessera in cui sono memorizzati in maniera gerarchica i nomi di tutti i DF ed EF trovati

1 Trattato in 4.4.3.
2 Trattato in 4.4.2.

- Load FileSystem: 

  viene lanciata la funzione che legge, se presente, il record nella carta in cui sono registrati tutti i file, e costruisce l'albero a partire da esso

- Create in Record: 

  la funzione chiamata da questo bottone visualizza un prompt per la creazione[1] e crea[2] il file nel percorso specificato, inoltre modifica il record del FileSystem aggiungendo il nome del nuovo file

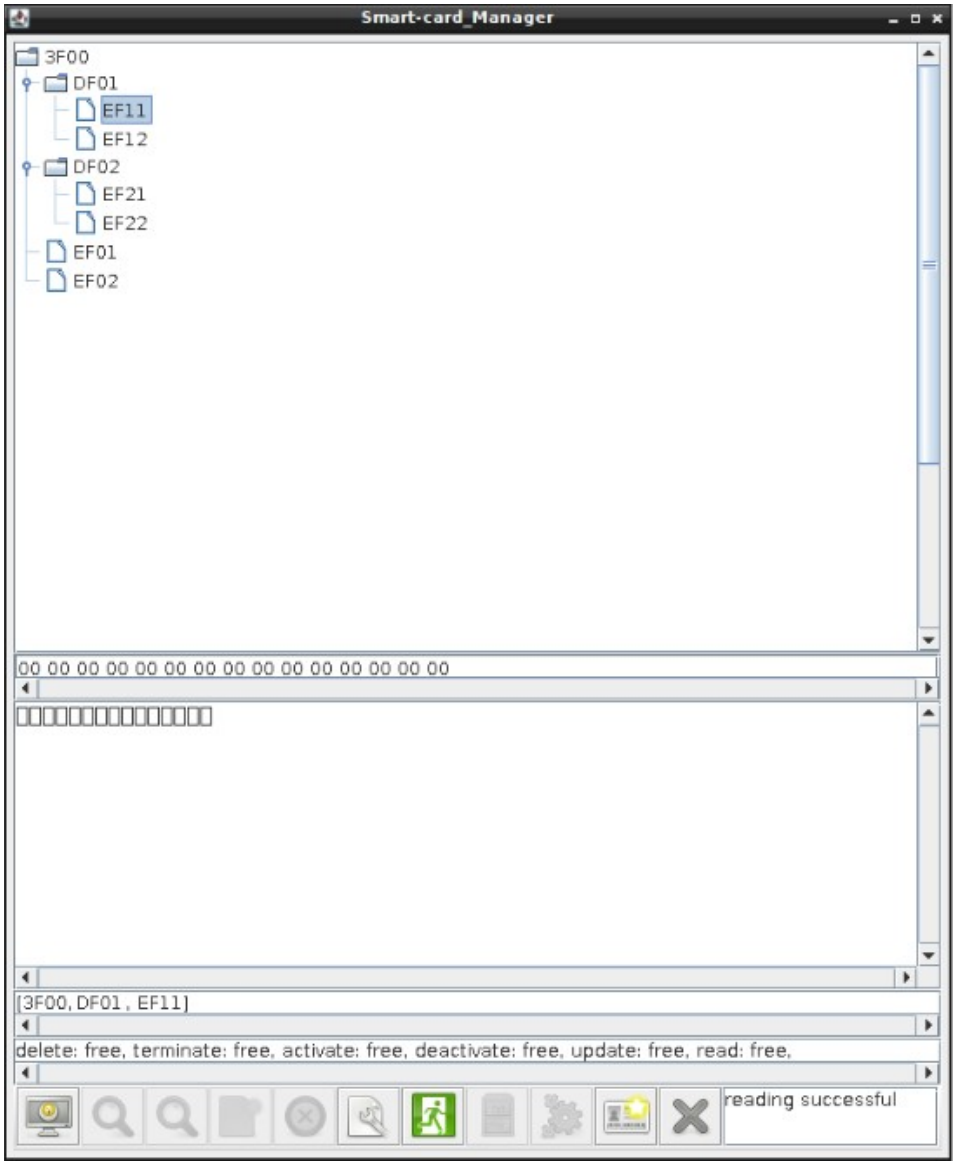

figura 11: lettura del file EF11 appena creato

1 Come in Figura 9
2 Trattato in 3.3.3.

- Destroy in Record: 

  questo lanciatore chiama la funzione di cancellazione[1] di un file ed aggiorna il record del FileSystem nella carta

Infine, slegato da qualsiasi altra funzione, c'è il tasto di uscita:

- Exit: 

  la sessione viene interrotta, il canale di comunicazione con la carta disabilitato e il programma e l'interfaccia grafica vengono chiusi

1 Trattato in 3.3.5.

4.4. Implementazione

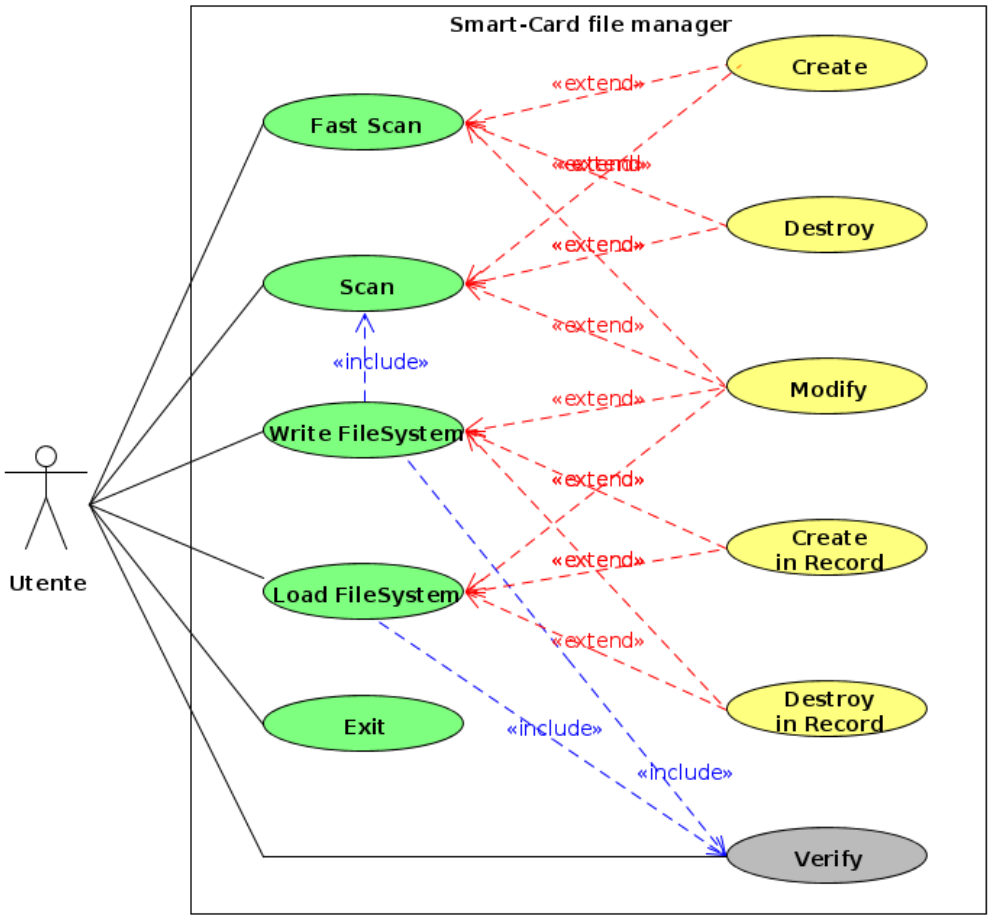

Figura 12: Diagramma dei casi d'uso

Il programma è stato pensato per offrire diverse funzioni per la gestione dei contenuti di una carta; prima di poter lavorare ed utilizzare un' utilità specifica, è però necessario disporre di una visione completa dei contenuti della tessera, tranne per il comando di autenticazione, per evitare fraintendimenti logici, come la creazione, non ammessa, di un file con lo stesso nome di un altro o come la selezione ambigua di un file, in una directory non conosciuta o che non è quella effettiva.

Per conoscere i nomi di tutti i file e la loro localizzazione all'interno della Smart-Card bisogna quindi aver creato l'ormai noto albero delle directory, con una funzione di scansione o tramite il record con il FileSystem; saltare questo passo precluderebbe il funzionamento dei comandi di creazione, cancellazione e modifica, e in generale renderebbe il progetto inutile. Se si ha scelto di creare l'albero con le

funzioni di scansione è possibile gestire i file nella carta ma senza che le modifiche di creazione e cancellazione vengano registrate nel record del FileSystem, per cui in una seguente sessione sarà necessario scansionare nuovamente la carta per avere a disposizione l'elenco veritiero dei file. Se invece si ha lanciato la funzione di Write FileSystem, oltre che avere a disposizione un albero aggiornato, si avrà aggiornato anche il record e sarà permesso utilizzare solamente i comandi concordi con questo utilizzo del programma, e cioè che aggiornano il record ogni volta che creano o eliminano file o directory. L'ideale sarebbe utilizzare questa funzione solamente per la prima sessione di utilizzo e seguitare con le successive lanciando sempre il comando Load FileSystem, che legge il record e genera l'albero di conseguenza, facendo risparmiare una notevole quantità di tempo, come descritto in seguito, rispetto alla scansione approfondita[1].

Figura 13: albero delle directory costruito

Una volta venuti a conoscenza della struttura dei contenuti si possono utilizzare i lanciatori delle funzione di gestione dei file indefinitamente e, nel caso della scansione approfondita o veloce, senza dover essere autenticati alla carta. Particolare attenzione va data a questo comando che non necessita un albero per agire in quanto non ha bisogno di conoscere la posizione di nessun file e quindi può essere lanciato prima di ogni altro; nonostante questo l'autenticazione è un prerequisito dei comandi con obblighi verso il record del FileSystem, che risulta così, relativamente più sicuro e meno tendente a cancellazioni involontarie.

1 Trattato in 4.4.2.

4.4.1. Gestione APDU

Fondamentale per la costruzione del progetto è l'utilizzo dei comandi APDU e l'interpretazione corretta dei messaggi di risposta, per ogni bottone e quindi funzione del programma:

- la scansione, di cui verrà parlato approfonditamente nel paragrafo successivo, sfrutta il comando SELECT FILE[1] per selezionare a ripetizione tutti i possibili file in una directory e posizionarli in maniera gerarchica nell'albero

- per la creazione di un file si usa principalmente il comando CREATE FILE[2]: quando appare il prompt di creazione, si selezionano le opzioni desiderate per il file da creare, e licenziando la finestra viene costruito tutto il comando, completo a seconda dei campi scelti per l'EF o il DF, e codificato in byte esadecimali comprensibili dalla carta; questi byte, costituiscono il campo data del corpo del comando e vanno accodati allo header prima di poter essere trasmessi, secondo quanto descritto in precedenza; ovviamente prima di creare un file bisogna selezionare la DF puntata dall'albero, ed il programma utilizza per questo scopo la selezione (SELECT FILE) per path assoluto a partire dal master file, distinguendo ciascun elemento per FID[3]

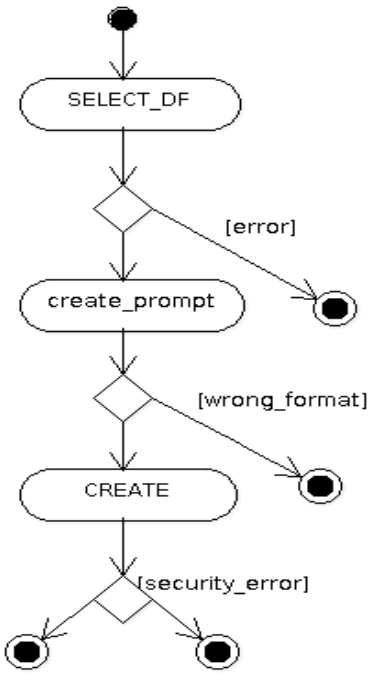

Figura 14: diagramma delle attività per la funzione di creazione

1 Trattato in 3.3.6.
2 Trattato in 3.3.3.
3 Trattato in 3.1.2.

- l'eliminazione di un file funziona inizialmente come la creazione: il file viene selezionato allo stesso modo per path assoluto, ma in questo caso viene poi chiamato il comando DELETE FILE[1], che senza particolari specifiche elimina l'elemento selezionato e ritorna i byte di stato risultato dell'operazione

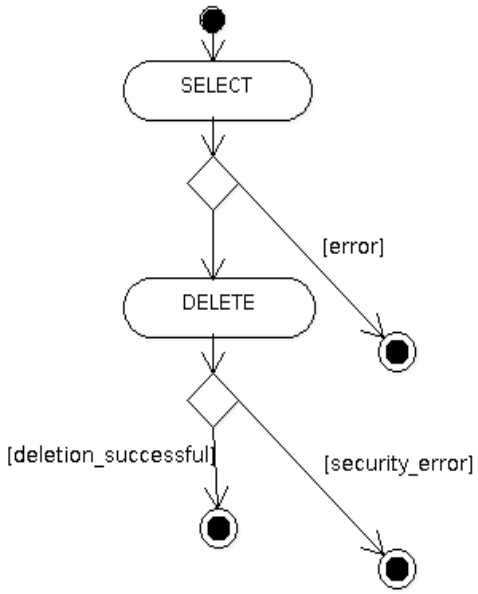

Figura 15: diagramma delle attività per la funzione di eliminazione

- la scrittura o modifica di un file, che utilizza il comando UPDATE BINARY[2], risulta più complessa delle precedenti: dopo l'usuale selezione per path assoluto, viene preso dall'area specifica il testo modificato e trasformato in sequenza di byte esadecimali; il comando standard non supporta la scrittura di un numero di byte maggiore a 0xFF, in quanto il campo Lc del comando stesso è codificato in un solo byte, per cui è necessario dividere i byte del testo da scrivere in blocchi da 255 byte, modificando l'offset per la scrittura di volta in volta incrementato appunto di 255; dopo aver scritto questi blocchi, si scrive la sequenza che contiene il resto dei byte dati dalla divisione fra la lunghezza del testo e 0xFF, considerando ovviamente sempre l'offset incrementato di 255, tutto secondo le specifiche del comando di scrittura descritte in precedenza. Risulta importante notare che non è possibile scrivere su un file un numero di byte maggiore a quelli per cui è stato creato, e qualsiasi stringa venga immessa nell'area di testo del contenuto da scrivere

1 Trattato in 3.3.5.
2 Trattato in 3.3.4.

verrà troncata alla lunghezza del file su cui si scrive. Ad esempio se si volesse scrivere una stringa di 1000 caratteri, cioè byte, in un file lungo 700 byte, saranno necessari:

- scrittura di 255 caratteri da 000 a 254 compresi, nel file con offset a 0
- scrittura di 255 caratteri da 255 a 509 compresi, nel file con offset a 255
- scrittura di 190 caratteri da 510 a 700 compresi, nel file con offset a 510
- i restanti caratteri da 701 a 1000 compresi, sono troncati e vengono persi

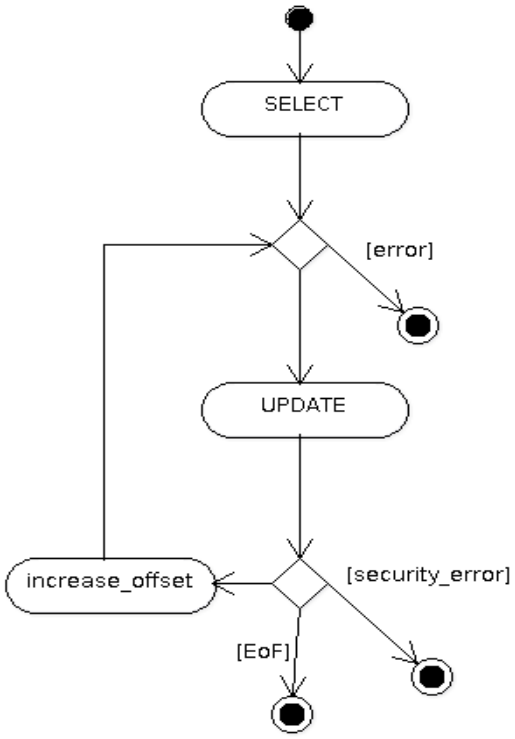

Figura 16: diagramma delle attività per la funzione di scrittura

- la lettura viene invece effettuata, diversamente da altri comandi, alla selezione di un elemento dell'albero delle directory, per mezzo del comando prima di selezione per path assoluto e poi di READ BINARY[1]; contrariamente al comando di scrittura, la lettura ha una portata massimo di 0xE7 byte, per cui la funzione deve leggere di volta in volta 231 byte incrementando l'offset di lettura di 0xE7 e concatenando le sequenze di byte successive in un unico vettore, che verrà poi tradotto anche in stringa Unicode visualizzata nella specifica area; diversamente da prima, se è rimasto un blocco da meno di 231 byte, il comando lo legge senza problemi ed alla successiva iterata, il ciclo si fermerà per ricevuto status byte diverso da "lettura effettuata", codificato con 0x9000

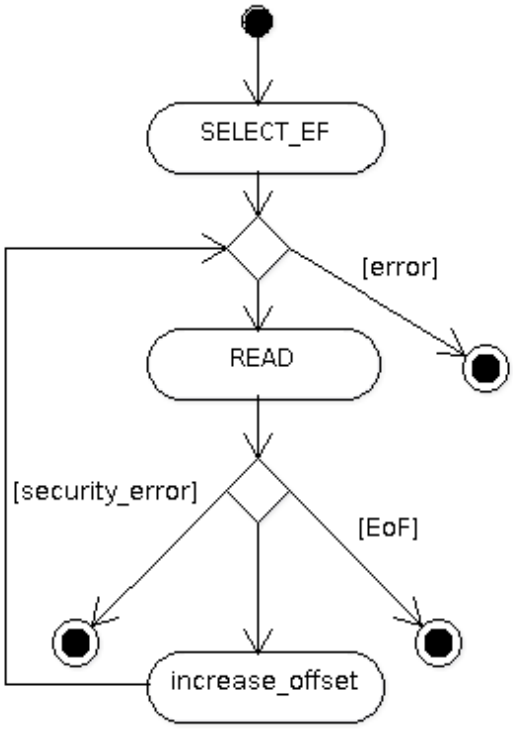

Figura 17: diagramma delle attività per la funzione di lettura

1 Trattato in 3.3.2.

- il comando di autenticazione lanciato dal bottone Verify, permette all'utente di autentificarsi globalmente alla carta tramite l'omonimo comando VERIFY[1]; il messaggio APDU è formato dallo header con il parametro settato per l'autenticazione globale come descritto in precedenza e dal corpo con il campo dati che contiene il PIN ottenuto con il form di immissione; questo comando non ha bisogno di ulteriori specifiche, ne selezioni di file a monte

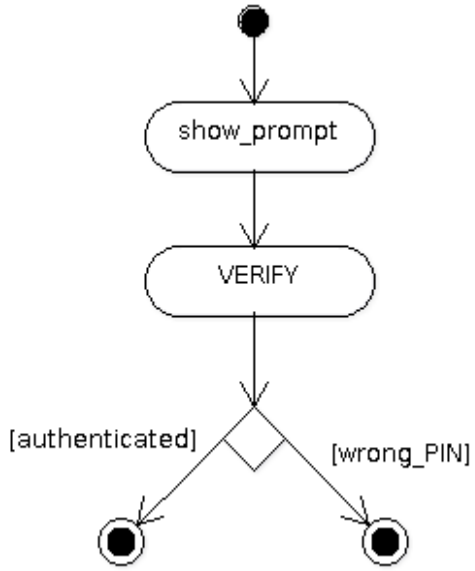

Figura 18: diagramma delle attività per la funzione di autenticazione

1 Trattato in 3.3.1.

- il lanciatore Write FileSystem utilizza un numero di comandi APDU maggiore rispetto agli altri: innanzi tutto chiama il comando di autenticazione utente, per poter lavorare sul record in maniera protetta, quindi attua una scansione approfondita della carta, generando l'albero e ottenendo una stringa formattata[1] in maniera particolare, e seleziona il file che lo contiene per poi eliminarlo, se esiste; tutto questo sfruttando sempre gli usuali comandi descritti nei paragrafi precedenti, di autenticazione, selezione e cancellazione; crea quindi un nuovo record di lunghezza pari a quella della stringa ottenuta dalla scansione, ed infine scrive la stringa con il FileSystem nel nuovo record

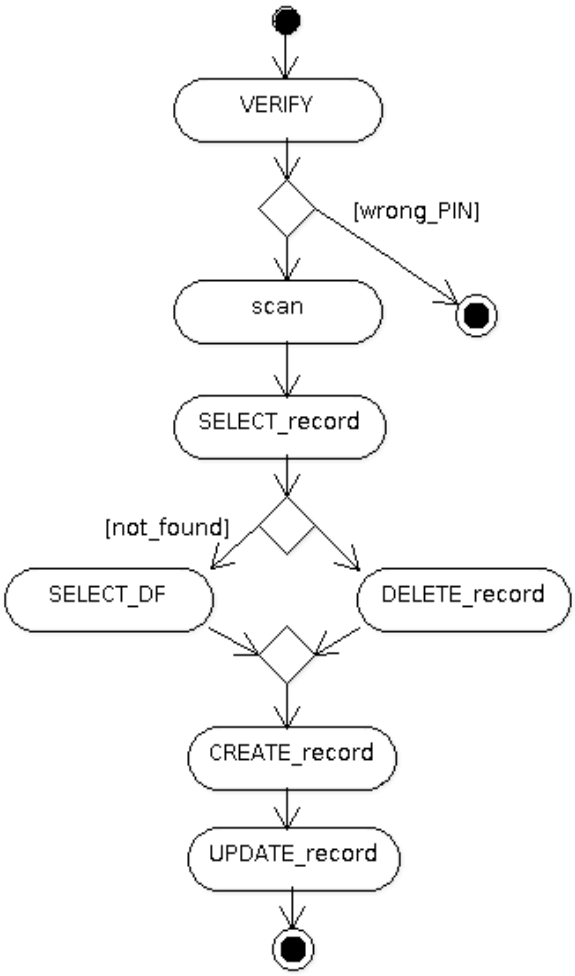

Figura 19: diagramma delle attività per la funzione di scrittura record

1 Trattato in 4.4.2.

- il lanciatore Load FileSystem ottiene la stringa con il FileSystem codificato, menzionata in precedenza, tramite una lettura al record di quest'ultimo, dopo di che lancia una scansione modificata che seleziona solo i file presenti nel record e nell'ordine gerarchico riscontrato in esso; delle modalità di scansione verrà parlato nel prossimo paragrafo

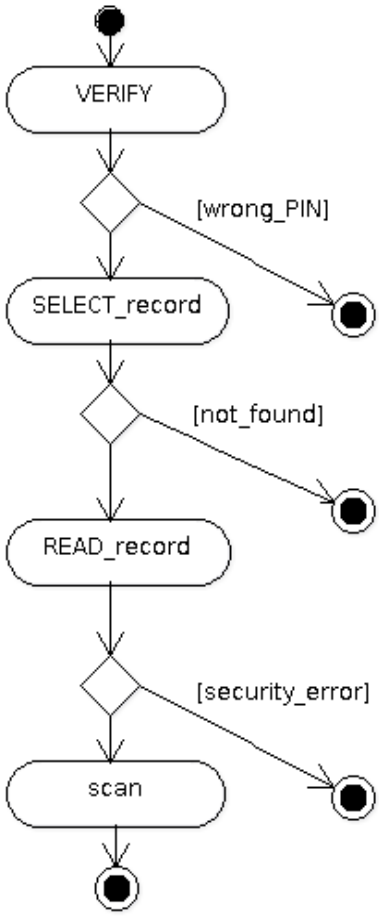

Figura 20: diagramma delle attività per la funzione di caricamento del record

- infine, i bottoni create in record e destroy in record, svolgono le funzioni di creazione ed eliminazione con un' ottica orientata al mantenimento aggiornato del record del FileSystem: se la funzione di base è andata a buon fine, il record viene aggiornato con le funzioni in sequenza, descritte nella parte di scrittura FileSystem: selezione / cancellazione / selezione / creazione / modifica del record, con una stringa in cui all'evenienza è stato aggiunto o tolto il nome dell'EF o della DF selezionati prima di premere il bottone, sempre in accordo con la logica in cui si è deciso di scrivere questo record[1].

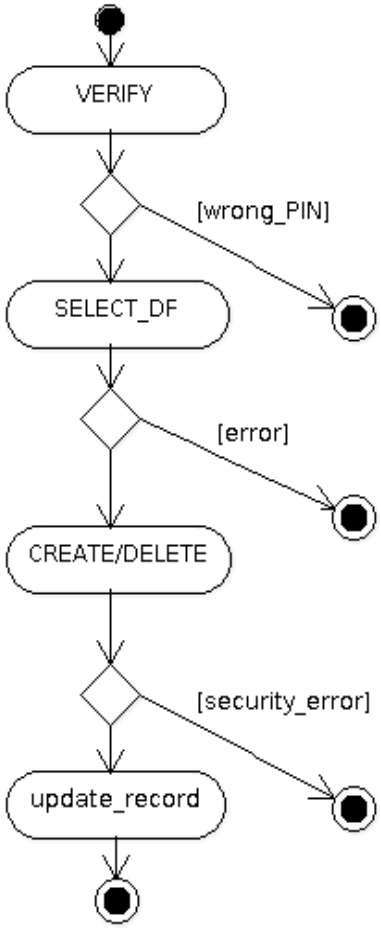

Figura 21: diagramma delle attività per la funzione di creazione ed eliminazione dal record

1 Trattato in 4.4.3.

4.4.2. Scansione

Per la corretta analisi, rappresentazione, e gestione di una Smart-Card, è d'obbligo una approfondita conoscenza dei contenuti, in termini di nome e posizionamento gerarchico di EF e DF nell'albero, di essa, per cui è stato necessario implementare una funzione di scansione della carta e di tutti i suoi file, non essendo disponibile un comando che ne facesse da surrogato.

L'idea è quella di muoversi all'interno del Master File e delle sue sotto directory verificando la presenza dei vari file; questo è reso possibile dall'ormai noto comando di selezione SELECT, che viene qui usato per scandagliare tutto lo spazio di nomi FID[1], all'interno di una singola directory; racchiudendo questo metodo di scansione in una funzione ricorsiva si può avere una mappa veritiera di tutti i contenuti della carta.

La funzione di scansione lanciata sul Master File, tenta di selezionare tutti i nomi dei possibili file contenuti in esso, ciclando su uno spazio dei nomi che va da 0x0000 a 0xFFFF, per un totale di 65536 potenziali nomi di file, per ogni cartella[2] ed incrementando di una unità il nome del file, visto come numero esadecimale nell'iterazione; è giusto notare che è stata scelta la modalità di selezione per path assoluto, per non creare errori di costruzione, causati dalla non univocità globale dei FID[3]. Durante la selezione esaustiva dei nomi, per quelli che hanno riscontro positivo, cioè degli status byte che codificano 0x9000, quindi selezione con successo, viene aggiunto graficamente un nodo figlio al master file nell'albero delle directory e, a quelli che inoltre risultano essere DF dall'attributo di categoria file del FCP[4], viene ammesso di contenere nodi figlio. A questo punto entra in gioco la ricorsività della funzione che, chiama se stessa sul nodo DF appena trovato e aggancia ad esso tutti i suoi nodi figlio, costruendo così una disposizione logica e gerarchica dei file, a cui si appoggia anche il comando di lettura, e quindi di selezione, che ricava il percorso assoluto del file dalla posizione grafica del nodo che lo rappresenta, in maniera

1 Trattato in 3.1.2.
2 Esclusi i nomi riservati, descritti in 3.1.2.
3 Ammessa in generale ma non per l' applicativo
4 Trattato in 3.1.3.

univoca.

Ovviamente tutto questo procedimento iterativo, per la scansione esaustiva dei nomi, e ricorsivo, per la chiamata a se stessa della funzione di scansione, comporta una notevole quantità di tempo, essendo necessario scandire tutti i nomi possibili all'interno di ogni DF presente nella carta. Proprio per questo la funzione di scansione può essere lanciata con alcune varianti che potrebbero o meno essere vantaggiose in termini di tempo.

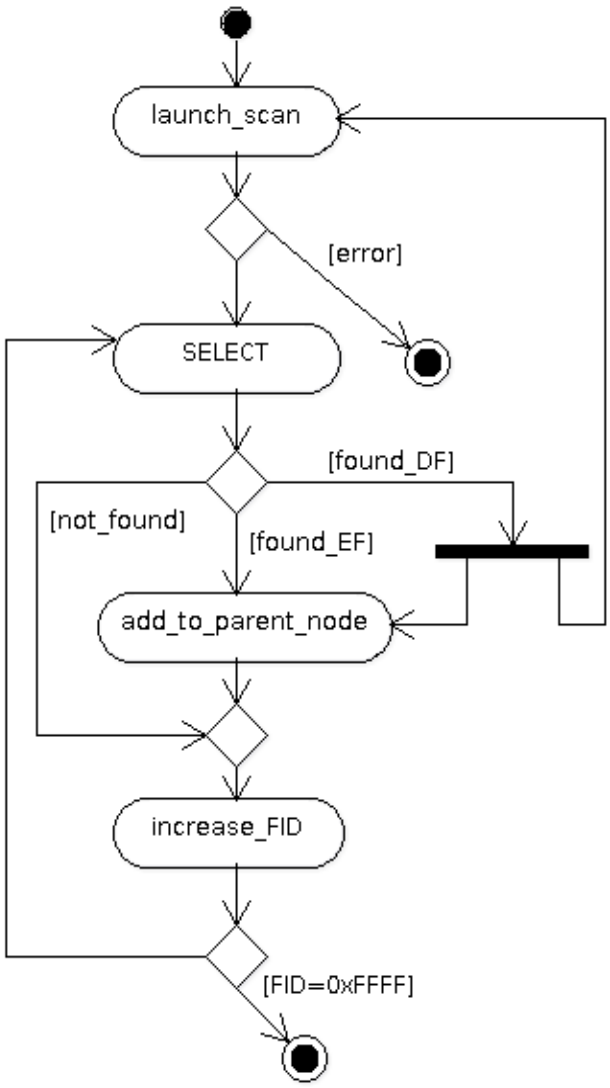

Figura 22: diagramma delle attività per la funzione di scansione

La scansione veloce è stata implementata a posteriori, in quanto presuppone una completa conoscenza dei contenuti della carta e di come sono disposti; questa modalità è nello specifico efficace grazie ad una funzione aggiuntiva che, ad ogni iterata, prende in input i due indici che rappresentano il FID che sta per essere selezionato e lo incrementano fino ad arrivare al prossimo indice che si sa con certezza essere presente all'interno della carta; questi "salti" permettono di selezionare solo i file noti evitando di dover scandagliare l'intero spazio dei nomi,

anche se questa concezione di scansione è stata possibile solo dopo una iniziale scansione approfondita della carta. Il lato poco utile di questa variante è che in ogni directory si tenta di selezionare tutti i file presenti nella carta, per non rendere troppo specifica la funzione.

La seconda variante è il tipo di scansione utilizzata dal lanciatore Load FileSystem, che non è una vera e propria scansione in quanto costruisce l'albero selezionando i nomi dei file presenti nel record con l'elenco gerarchico delle directory. Come per la scansione veloce, questo è reso possibile grazie ad una funzione che dati gli indici che descrivono l'ultimo FID selezionato, trova nel record il successivo nodo da aggiungere, tenendo conto della struttura gerarchica delle DF, memorizzata nel file che contiene il record. Questa funzione è particolarmente importante perché alla base dell'idea di FileSystem in una Smart-Card, e permette quindi un utilizzo immediato delle funzioni del software, portando a conoscenza dell'utente tutti i contenuti della carta.

4.4.3. File System

Come descritto in precedenza, per accedere ai contenuti di una carta è necessario conoscere il FID di ogni elemento o per lo meno la posizione all'interno della gerarchia degli EF e DF, per la selezione ed in seguito lettura o scrittura; una scansione di tutti i file era necessaria per avere un quadro completo ma con questa ottica era obbligatorio ripetere la procedura ad ogni sessione di utilizzo, rendendo abbastanza scomodo l'utilizzo del programma, ed inoltre, anche con una scansione veloce, conoscendo a priori i FID di ogni file, quei file che venivano creati od eliminati in una sessione, non erano considerati nella successiva, a meno di modificare ogni volta il codice, prima dell'esecuzione; questo, oltre che poco pratico, va anche contro l'idea di un tool user-friendly ad alto livello di astrazione, per cui è sembrato necessario registrare tutti i dati relativi alle relazioni fra i vari file e directory della carta, proprio in un file su di essa, in modo da poterlo leggere e velocemente ricreare l'albero delle directory, che viene comunque aggiornato ad ogni creazione ed eliminazione avvenuta nella Smart-Card. A questo punto, mantenere un mero elenco di nomi dei file era riduttivo, quindi si è deciso di organizzarli nel record in maniera gerarchica, con caratteri che quantificano ogni elemento:

<3F00<DF01<EF11><EF12>><DF02<EF21><EF22>><EF01><EF02>>

secondo:

- i FID sono composti da 4 caratteri[1]
- '<' è posto prima di ogni FID e distingue un file dal precedentemente
- '>' è posto dopo ogni FID e distingue il file dal successivo
- se dopo un file, invece di '>' è presente un altro '<' che indica l'inizio di un nuovo file, il primo è una DF e contiene tutti i nomi dei file che sono presenti prima del suo carattere di chiusura '>'

1 Trattato in 3.1.2.

In maniera grafica e indentando il record del FileSystem è facile notare una netta somiglianza con l'usuale albero delle directory della GUI:

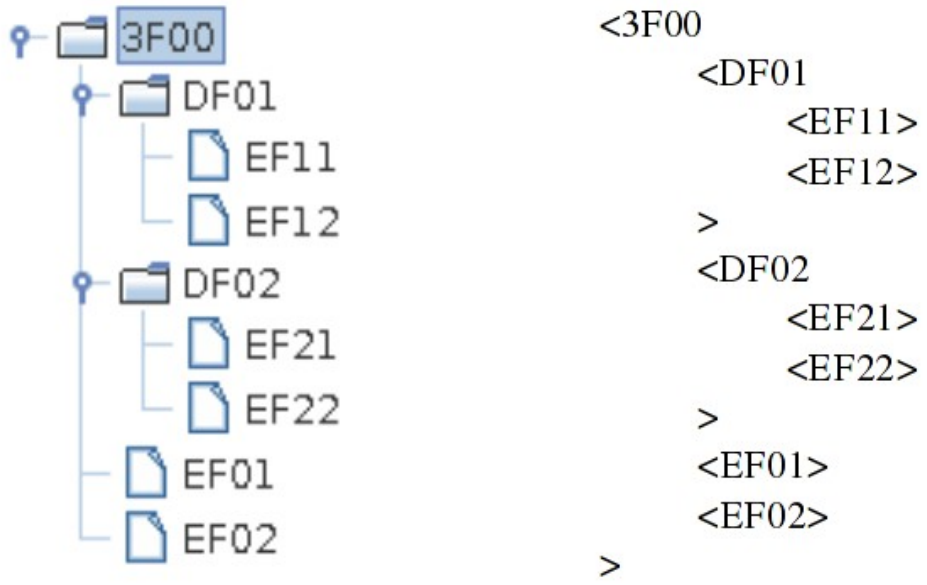

Figura 23: record FileSystem ed albero delle directory a confronto

5. Sviluppi futuri

Nonostante l'applicativo risulti completo dal punto di vista dello scopo primario del progetto, che è quello di gestire file a livello utente, le possibilità di ampliamento e miglioramento sono molteplici, in quanto le caratteristiche della carta Enjoy My Unicam, e in generale di qualsiasi altra Smart-Card, non sono sfruttate appieno; l'obiettivo del software era limitato ai file semplici ed una prima idea sarebbe quella di gestire anche i file di sicurezza, cioè PIN, password e chiavi crittografiche, in termini di creazione, modifica ed eliminazione, coinvolgendo anche tutte le funzioni specifiche della carta, relative all'infrastruttura a chiave pubblica PKI, come quelle di cifratura, sia a chiave pubblica che privata, di firma digitale ed autenticazione certificato e quelle di hash[6].

Alcuni dei comandi che potrebbero essere implementati con più facilità, sono quelli per la gestione del PIN, come il già noto VERIFY[1], cui potrebbero essere affiancati i comandi CHANGE REFERENCE DATA e RESET RETRY COUNTER, rispettivamente per la modifica del PIN e per il reset del contatore di tentativi di immissione; inoltre si potrebbe pensare di gestire i file protetti, con un PIN diverso da quello globale della carta, attraverso gli attributi dei file stessi.

Infine si potrebbero considerare anche tutte quelle parti che per motivi di tempo o di efficienza sono state tralasciate, come l'utilizzo di file di tipo diverso da Transparent Elementary File[2], la codifica di attributi di sicurezza multipli per un singolo elemento[3], e l'implementazione delle funzioni di modifica dello status operativo di DF ed EF[4].

In generale, per ovvi motivi, non si è potuto codificare tutte le funzioni della carta nei dettagli, e quindi un ottimo sviluppo futuro potrebbe essere quello di sfruttare al massimo le enormi potenzialità della Smart-Card.

1 Trattato in 3.3.1.
2 Trattato in 3.1.1.
3 Menzionato in 3.1.4.
4 Figura 3

6. Conclusioni

Nonostante attualmente le applicazioni e le modalità di utilizzo delle schede a circuito integrato siano già ampie, basti pensare alle transazioni bancarie, acquisti in generale ed il concetto di firma digitale su cui sono state costruite, lo sviluppo futuro potrebbe comprendere un molteplice sfruttamento di tutte quelle che sono le caratteristiche operative delle carte, non ancora del tutto adoperate. Già riassumendo in una singola scheda tutte quelle che la persona media tiene nel proprio portafoglio, si avrebbe un guadagno, perlomeno in termini di spazio: carta d'identità, tessera sanitaria, patente di guida, carta di credito, tessere varie di abbonamenti ai trasporti, al cinema, alla palestra, tessere identificative dei servizi e all'interno delle aziende, e molto altro.

L'utilizzo principale dell'applicativo prodotto, oltre che quello accademico, è ricondotto alla registrazione di qualsiasi tipo di dato personale, per gli scopi più vari; un' attività di qualsiasi tipo che abbia bisogno di interagire con gli utenti in maniera sicura, potrebbe creare un file nella tessera personale dei suoi clienti e registrarne i dati personali; in questo modo gli utenti si potrebbero ritrovare con una unica Smart-Card che contiene tutti i loro dati per i più diversi scopi, in quanto ad ogni servizio di cui fanno uso, sarebbe dedicata solo una piccola porzione della loro tessera, invece che avere una scheda per ogni servizio, abbonamento o utilizzo, semi vuota e poco sfruttata, che va a contribuire allo spessore del porta documenti che chiunque possiede.

Oltre a questo, un altro utilizzo ipotetico sarebbe quello di serbatoio di informazioni personali, come ad esempio una storia medica completa, comprensiva di allergie, terapie, sensibilità ai farmaci e operazioni chirurgiche affrontate, in modo da essere prontamente consultata in caso di ricovero, senza dover ricorrere agli archivi della specifica clinica che ha seguito il paziente in precedenza.

Le possibilità sono quindi molte, ma molto spesso poco sfruttate, anche se in alcuni casi, come quello della tessera d'identità malese MyKad, vengono ampiamente coperte, e vanno dalla ovvia identificazione, che comunque comprende impronte

digitali e fotografia, alla patente, passando per documento di viaggio, tessera dei trasporti, cartella clinica, portafoglio elettronico, firma digitale, tessera bancaria e molto altro.

7. Ringraziamenti

Prima di tutti voglio ringraziare la mia famiglia, che nonostante tutto ha sempre creduto che potessi portare a termine questo percorso di studi, e mi ha supportato in questi anni, nelle mie decisioni non sempre giuste

devo poi ringraziare tutti gli amici, vecchi e nuovi, per le piacevoli giornate insieme, nello studio e nel divertimento,

grazie al corpo docenti, che non ha mancato nel tenere alto il mio interesse per la disciplina,

infine vorrei ringraziare il mio Relatore per l'impegno e la correttezza e Damiano per la molta pazienza ed i consigli tecnici.

Grazie a Tutti

8. Sitografia

1. http://www.pcsistemi.it/assistenza/documenti/46-generale/158-isoiec-78102003-iso-7810
2. http://www.smartcardbasics.com/smart-card-types.html
3. http://www.smartcardalliance.org/pages/smart-cards-faq
4. http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816-4_5_basic_organizations.aspx
5. http://archivio.digitpa.gov.it/sites/default/files/formazione/SmartCardAvanzato18062004_0_0.pdf
6. http://www.acsiel.fr/iso_album/ias_ecc_v1_0_1uk.pdf
7. http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816-3.aspx
8. http://www.tml.tkk.fi/Studies/T-110.497/2003/lecture4.pdf?q=apdu
9. http://www.hidglobal.com/products/readers/omnikey/5321
10. http://www.bpa.it/page/enjoy-my-unicam
11. http://www.namirial.com
12. http://docs.oracle.com/javase/7/docs/jre/api/security/smartcardio/spec/javax/smartcardio/package-summary.html
13. http://ludovic.rousseau.free.fr/softwares/pcsc-tools/
14. http://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html
15. http://docs.oracle.com/javase/7/docs/api/java/awt/package-summary.html
16. http://www.primianotucci.com/os/smartcard-explorer
17. http://www.cardwerk.com/smartcards/smartcard_history.aspx
18. http://www.zhenghuacard.com/en/index.html
19. http://www.jpn.com.my/docs/MyKad.htm