

LA SMART CARO

Tesina realizzata da Micheli Mirko

Classe 5[^]BZ

Anno Scolastico 2004/2005



Sommario

<u>INTRODUZIONE</u>	4
---------------------------	---

La Smart Card

<u>1.2 ARCHITETTURA HW/SW DI UN SISTEMA BASATO SU SMART CARD</u>	5
<u>1.3 LA SMART CARD NELLA CRITOGRAFIA</u>	6
<u>1.4 LA SMART CARD NELLE PROCEDURE DI AUTENTICAZIONE</u>	7
<u>1.5 LE APPLICAZIONI DELLA SMART CARD</u>	8
<u>1.6 GLI STANDARD</u>	10
<u>1.7 CORPORATE CARD</u>	11
<u>1.7.1 Carta Multiservizi della Difesa</u>	12

Fondamenti tecnologici

<u>2.1 LA SMART CARD</u>	14
<u>2.2 MEMORY CARD</u>	15
<u>2.3 MICROPROCESSOR CARD</u>	15
<u>2.4 SMART CARD CONTACT, CONTACTLESS E DUAL INTERFACE</u>	16
<u>2.1.4 Token USB</u>	17
<u>2.2 Architettura del microchip</u>	18
<u>2.6 PRODUZIONE DELLE SMART CARD</u>	20
<u>2.7 IL SISTEMA OPERATIVO</u>	20
<u>2.8 PROTEZIONE DEI DATI SULLA SMART CARD</u>	21
<u>2.9 QUALE SMART CARD?</u>	22

Lo standard ISO 7816

<u>3.1 LE SPECIFICHE TECNICHE STANDARD</u>	23
<u>3.2 CARATTERISTICHE FISICHE ED ELETTRICHE</u>	24
<u>3.3 L'ATR</u>	26
<u>3.4 PROTOCOLLI DI TRASMISSIONE</u>	26
<u>3.4.1 Protocollo T = 0</u>	27
<u>3.4.2 Protocollo T = 1</u>	27
<u>3.5 STRUTTURA E FORMATO DEI DATI MEMORIZZATI NELLA EEPROM</u>	27
<u>3.5.1 File System</u>	28
<u>3.5.2 Tipologia e formato di un file</u>	29
<u>3.5.3 Permessi di accesso ai file</u>	30
<u>3.5.4 Secure Messaging</u>	30
<u>3.6.1 Command APDU</u>	31
<u>3.6.2 Response APDU</u>	31
<u>3.7 CICLO DI VITA</u>	33

Implementazioni

<u>4.1 LE SMART CARD SULLE PIATTAFORME MICROSOFT</u>	34
<u>4.1.1 CryptoAPI</u>	34
<u>4.1.1.1 Internet Explorer</u>	35
<u>4.1.1.2 Outlook</u>	36
<u>4.1.1.3 Smart card logon su Windows 2000/XP</u>	36
<u>4.2 LA PROCEDURA DI ACCESSO AL SISTEMA</u>	36
<u>4.3 ACCESSO AI SISTEMI WINDOWS 2000 CN SMART CARD</u>	37
<u>4.3.1 Configurare il dominio Windows 2000</u>	38
<u>4.3.2 Configurare Certificate Server</u>	38
<u>4.3.3 Specificare la policy</u>	39
<u>4.3.4 Configurare l'Enrollment Station</u>	39
<u>4.3.5 Emissione dei certificati su smart card</u>	40

Crittografia

<u>5.1 LA CRITTOGRAFIA</u>	40
<u>5.2 ALGORITMI A CHIAVE PRIVATA</u>	43
<u>5.3 ALGORITMI A CHIAVE PUBBLICA</u>	44
<u>5.4 LA TECNICA ADOTTATA NELLA PRATICA</u>	45
<u>5.5 LA FIRMA DIGITALE</u>	46
<u>5.6 GLI ALGORITMI DI HASHING</u>	46
<u>5.7 LA CERTIFICAZIONE</u>	47

PARTE PRATICA

<u>SCHEMA ELETTRICO LETTORE SMART CARD</u>	49
<u>CIRCUITO SU BASETTA</u>	49
<u>Lato Pise:</u>	49
<u>Lato componenti:</u>	49
<u>COMPONENTISTICA</u>	50
<u>RELAZIONE</u>	50
<u>Programma PIC16F628:</u>	51
<u>Borland Delphi 5:</u>	84

INGLESE

<u>HISTORY</u>	88
<u>CONTACT SMARTCARD</u>	88
<u>CONTACTLESS SMARTCARD</u>	88
<u>APPLICATIONS</u>	89
<u>CURRICULUM VITAE</u>	89

INTRODUZIONE

La smart Card è un dispositivo hardware, delle dimensioni di una carta di credito che si propone in primo luogo come dispositivo di elaborazione a supporto della crittografia, ed in grado di proteggere e memorizzare chiavi crittografiche private.

C'è inoltre da considerare anche la sua capacità di memorizzare ed elaborare dati al suo interno, non che le sue ridotte dimensioni che permettono all'individuo di poter usufruire e portar sempre con se questo dispositivo.

Negli ultimi cinque anni, le tecnologie basate su smart card hanno ricevuto molte attenzioni in ambito informatico e nel campo delle telecomunicazioni, grazie soprattutto alla capacità di questo dispositivo di conservare dati in maniera estremamente sicura. Questa caratteristica ha consentito di realizzare applicazioni sia nell'ambito della sicurezza informatica (in particolare nell'identificazione automatica dell'individuo e nella sua certificazione) sia in altri ambiti: telecomunicazioni mobili, PayTV, commercio elettronico, sistemi bancari, SIM card (in ambito GSM), Carta d'Identità Elettronica, carta di pagamento bancaria, BADGE aziendale, carte prepagate e di raccolta punti. Sono solo alcune delle possibili applicazioni realizzate con la smart card che sta ormai diventando uno strumento indispensabile nella vita quotidiana.

Sebbene la tecnologia sia ben consolidata, le numerose applicazioni della smart card e le differenti caratteristiche delle piattaforme di elaborazione, hanno portato alla definizione di numerosi standard e/o specifiche tecniche che mirano a definire una piattaforma comune che consenta l'interoperabilità tra smart card di diversa fabbricazione.

Lo scopo di questa mia tesina è quello di illustrare in maniera sintetica, ma esauriente gli standard, le caratteristiche e le tecnologie in ambito smart card, fornendo dapprima una visione generale dello scenario tecnologico e, successivamente, un approfondimento tecnico, rivolto principalmente al programmatore, sulle architetture, i paradigmi di programmazione e le API relative a ciascuno standard e/o specifica tecnica.

La prima parte della tesina introduce la tecnologia delle smart card, i concetti generali ed alcuni *case study*.

La seconda parte, invece, è rivolta alle diverse specifiche tecniche e ai paradigmi di programmazione proposti in ambito smart Card.

La Smart Card

1.1 LA STORIA:

L'idea di incapsulare un circuito integrato in un supporto di plastica fu introdotta nel 1968 da due inventori tedeschi: Jurgen Dethloff e Helmut Grotrupp. Negli anni '70 furono depositati i primi brevetti da parte di diverse aziende e gruppi di ricerca, ma solo alle soglie degli anni '80 Bull (allora CII-Honeywell-Bull) mise in commercio il primo prototipo di smart card e introdusse le smart card a microprocessore.

Le prime applicazioni con smart card furono realizzate in Francia e Germania all'inizio degli anni '80 dove le smart card furono adoperate come carte telefoniche prepagate e come carte bancarie di credito/debito ad alta sicurezza. Tali applicazioni mostrarono la grande capacità di resistenza ad attacchi e la considerevole flessibilità delle smart card e traghettarono la nuova e vincente tecnologia verso i recenti sviluppi in ambito GSM e Web.

Negli ultimi anni le nuove tecniche di miniaturizzazione, mediante le quali è stato possibile produrre microchip sempre più piccoli a costi sempre più bassi, hanno consentito di realizzare smart card più potenti dotate di coprocessore crittografico e di buone capacità di memoria a costi accessibili. Tale disponibilità ha avviato una fase di notevole e sorprendente sviluppo che è partita dall'implementazione delle SIM card in ambito **GSM** fino ad arrivare alla realizzazione della Carta d'Identità Elettronica e delle carte di credito "intelligenti".

1.2 ARCHITETTURA HW/SW DI UN SISTEMA BASATO SU SMART CARD

La figura sotto riportata mostra l'architettura hardware di un sintetico sistema di elaborazione che consente l'utilizzo di smart card. Oltre a tastiera e mouse, al personal computer è collegato un terminale di lettura (che si può vedere nella pagina seguente al di sopra del mouse) detto tipicamente lettore di smart card che consente di comunicare con la smart card. La maggior parte dei lettori in commercio possono essere collegati al PC tramite porta seriale o USB. Per i computer portatili sono anche disponibili lettori **PCMCIA**.

Esistono inoltre (anche se meno diffusi) lettori collegati alla tastiera, o montati su floppy disk. Le funzionalità tipiche di un lettore di smart card sono:

- fornire l'alimentazione e il segnale di clock al microchip della smart card;

- gestire il canale di I/O mediante il quale sono scambiati i dati digitali da e per la smart Card.



Un'applicazione software con smart card consiste in un'applicazione in esecuzione sul computer che interagisce con la smart card mediante le funzionalità offerte dal circuito integrato. Lo schema sotto riportato mostra l'architettura logica di un sistema software che dialoga con la smart card.

A partire dal basso, il lettore di smart card si interfaccia con il Sistema Operativo del computer attraverso un driver. Il Sistema Operativo mette a disposizione un insieme di funzioni a basso livello per comunicare con il lettore che possono essere usate direttamente dalle applicazioni o possono essere mappate in un insieme di funzioni ad alto livello (implementate in un *middleware*), che fornisce pertanto un'interfaccia di programmazione più semplice e flessibile per scrivere applicazioni.

1.3 LA SMART CARD NELLA CRITOGRAFIA

La crittografia, sebbene fornisca delle tecniche estremamente eleganti e potenti per assicurare l'integrità e la riservatezza delle informazioni, registra un punto debole: la difficoltà di proteggere in maniera adeguata le chiavi private. Difatti, se la chiave privata adottata per cifrare un insieme di informazioni non è adeguatamente protetta (ad esempio è memorizzata sull'hard disk di un PC), un malintenzionato potrebbe impossessarsene e quindi decifrare tutte le informazioni vanificando gli sforzi crittografici compiuti.

La smart card a microprocessore, grazie alle caratteristiche di protezione dei dati intrinseche del microchip e alla presenza di un coprocessore crittografico che gli consente di eseguire le principali funzioni crittografiche on-board, (senza esporre le chiavi private all'ambiente operativo delle applicazioni nel quale potrebbero essere attaccate da programmi ostili) si propone come mezzo adeguato a proteggere le chiavi private rilanciando la crittografia come supporto tecnologico di base per lo sviluppo di sistemi informatici sicuri e riproponendo in maniera decisa la firma digitale come sicuro e insostituibile strumento per l'autenticazione e l'identificazione degli individui, per la verifica dell'integrità di insiemi di dati e per il non ripudio delle transazioni.

Architettura logica di un sistema software che dialoga con una smart card



1.4 LA SMART CARD NELLE PROCEDURE DI AUTENTICAZIONE

Per le sue peculiarità, la smart card è fortemente usata nelle procedure di **Strong Authentication** come dispositivo di memorizzazione sicura delle caratteristiche relative all'utente. Tipicamente, tali credenziali sono rappresentate dalla coppia **username-password**, o, nei sistemi più avanzati basati su firma digitale, dalla coppia **chiave privata-certificato digitale**.

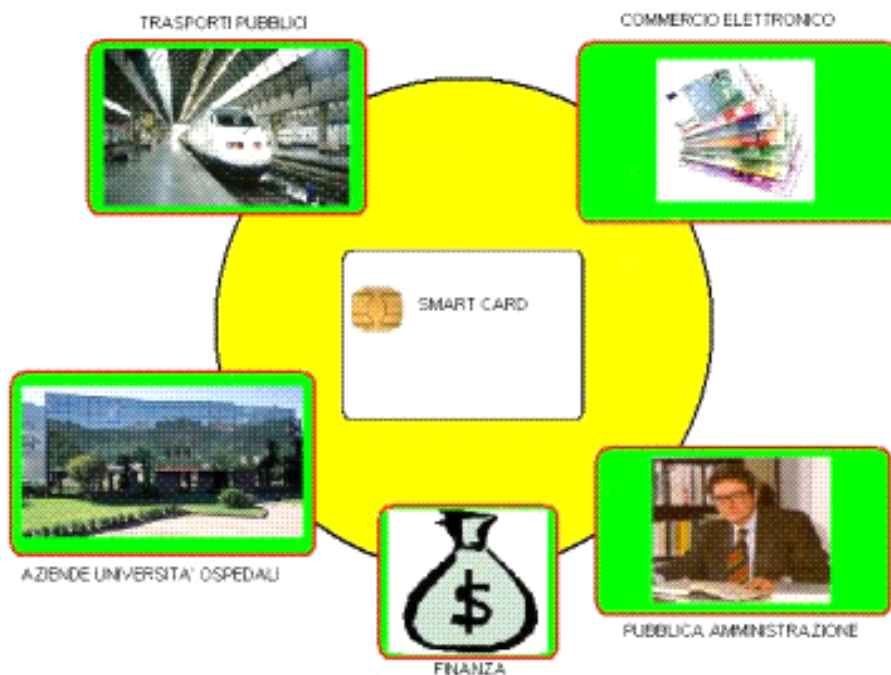
L'accesso alle credenziali è subordinato alla verifica di un PIN (**Personal Identification Number**) che il titolare della smart card è tenuto a conservare con cura e deve digitare durante la procedura di autenticazione quando richiesto. In seguito alla verifica positiva del PIN da parte della smart card, le credenziali utente vengono lette dalla procedura e usate per eseguire l'autenticazione. Nel primo caso la coppia **username-password** viene inviata al sistema che richiede l'autenticazione. Nel secondo caso, la chiave privata viene usata dal coprocessore crittografico della smart card per apporre una firma digitale che viene poi inviata al sistema che richiede l'autenticazione per l'opportuna verifica.

La sicurezza dei sistemi di **strong authentication** con smart card è basata sostanzialmente sul possesso della smart card e sulla conoscenza del PIN. Per innalzare ulteriormente il livello di sicurezza il PIN può essere sostituito da un'informazione biometrica, quale l'impronta

digitale, la conformazione dell'iride, ecc.. In questo modo l'accesso alle credenziali utente sulla smart card non è più subordinato alla verifica di un PIN (che potrebbe in qualche modo essere estorto) ma alla verifica dell'informazione biometrica del titolare memorizzata sulla smart card con un campione acquisito durante la procedura di autenticazione (mediante uno speciale dispositivo di acquisizione quale può essere uno scanner di impronta digitale). La soluzione con tecnologia biometrica ha due vantaggi: da un lato l'accesso alle credenziali utente sulla smart card è consentito solo in presenza del titolare (il possessore della smart card e dell'informazione biometrica), dall'altro il titolare non è tenuto a ricordare alcun PIN. Alcune smart card dell'ultima generazione supportano un'estensione della funzione di verifica dei PIN che consente di far eseguire direttamente al microprocessore l'algoritmo di corrispondenza specifico per l'impronta digitale. Con tale estensione, l'impronta originale del titolare non viene mai estratta dalla memoria smart card e pertanto assicura un elevatissimo livello di sicurezza e rispetta le normative vigenti in maniera di tutela della privacy.

1.5 LE APPLICAZIONI DELLA SMART CARD

- Le applicazioni con smart card coprono numerosi campi che vanno dal settore pubblico, a quello privato e quello delle telecomunicazioni (come illustrato nell'immagine sottostante). In ogni caso le smart card sono usate principalmente per la memorizzazione sicura dei dati e per autenticare e proteggere le transazioni che avvengono nella rete.



Di seguito sono descritte le principali applicazioni implementate mediante smart card:

- **Telefonia fissa**

In questo ambito la smart card è usata principalmente come carta prepagata grazie alle sue caratteristiche di inviolabilità.

- **Telefonia mobile**

Lo standard GSM ha introdotto il concetto di SIM (**Subscriber Identity Module**) che rappresenta un dispositivo portatile e di identificazione dell'utente. La SIM conserva le informazioni di identificazione degli utenti e genera le chiavi crittografiche usate per cifrare la trasmissione digitale della voce. Pertanto può essere inserita in cellulari diversi presentando al gestore telefonico sempre la stessa identità.

I requisiti funzionali delle SIM corrispondono perfettamente con le funzionalità offerte dalle sim-card e pertanto, l'implementazione di smart card (con supporto di plastica di dimensioni ridotte) è stata una scelta naturale. Analoghe considerazioni si applicano in ambito UNITS dove le USIM sono l'evoluzione delle SINI.

- **Bancario**

In ambito bancario le smart card sono state utilizzate soprattutto come **Borsellini elettronici**, **Denaro elettronico** e in alcuni casi come **carte di credito**. Considerando che un borsellino elettronico (deve consentire il caricamento, la memorizzazione e l'uso del credito evitando nello stesso tempo clonazioni e modifiche fraudolente del credito, le smart card rappresentano ancora una volta la soluzione ottimale che soddisfa tutti i requisiti. A conferma di ciò si ricordano le numerose implementazioni tra cui: VisaCash e Mondex in UK, Geldkarte in Germania, Minipay in Italia e Moneo in Francia.

- **E-government**

In quest'ambito le smart card sono state utilizzate principalmente come strumento di identificazione e di memorizzazione di informazioni personali. Tra le numerose realizzazioni si ricordano: **Carta di identità elettronica**, **Carta Nazionale Servizi**, **Carte sanitarie elettroniche** (health cards.), **Carte pensione** (social security card), Schede elettorali elettroniche, **Carte di firma digitale** a valore legale, ecc.

- **Trasporti**

Nei trasporti pubblici le smart card sostituiscono i biglietti e diventano a tutti gli effetti titoli di viaggio. Tipicamente sono utilizzate come biglietti prepagati o in sostituzione degli abbonamenti. Nei trasporti privati invece le smart card sono spesso utilizzate come carte di pagamento per i parcheggi pubblici

- **Militare**

In questo settore le smart card hanno consentito di realizzare la "carta del militare" che rappresenta sia un vero e proprio documento di riconoscimento del militare, sia un contenitore di informazioni personali tra cui informazioni sanitarie (estremamente utili in caso di ferimento durante un'operazione di guerra), informazioni relative al grado e al corpo di appartenenza, ecc.

- **Settore Privato**

In tale ambito, le principali necessità riguardano il controllo dell'accesso ad aree riservate, il controllo dell'accesso alle reti e ai sistemi di calcolo (autenticazione e **firewalling**) e la protezione delle postazioni di lavoro (tipicamente personal computer) mediante sistemi di **login** sicuro e cifratura dei dati sensibili.

- *Altre*

Mediante smart card sono state implementate una serie di altre applicazioni tra cui: applicazioni di fidelizzazione (*Loyalty System*) come le carte per la raccolta punti, applicazioni riguardanti la logistica e il tracciamento delle operazioni (*Fleet Management* e *Operation Tracking Systems*), applicazioni di movimentazione materiali e mezzi, ecc.

1.6 GLI STANDARD

Fino a qualche anno fa, le differenze esistenti tra le varie smart card disponibili sul mercato e tra i numerosi dispositivi di lettura rendevano estremamente complesso lo sviluppo di applicazioni che dovessero interagire con le smart card. A causa delle notevoli differenze tra le molteplici implementazioni native, solo un numero ristretto di programmatori altamente specializzati era in grado di scrivere applicazioni che funzionavano però solo con un insieme estremamente ristretto di smart card e/o lettori (tipicamente di un solo tipo).

Lo sviluppo tecnologico degli ultimi anni, unito alla domanda sempre più forte di applicazioni con smart card, ha portato alla definizione di un insieme di specifiche standard che consentisse l'interoperabilità tra smart card e lettori di diversa fabbricazione. Lo standard ISO 7816 definisce le caratteristiche fisiche, elettriche e funzionali che sono alla base di una smart card allo scopo di standardizzare sia le proprietà esterne della smart card (dimensioni, disposizione dei contatti elettrici, ecc) sia i protocolli di comunicazione tra il circuito integrato, il terminale di lettura e le applicazioni che sono eseguite sul computer. Tuttavia tale standard, sebbene da un lato definisce le fondamenta tecnologiche delle smart card, dall'altro propone un'interfaccia di programmazione veramente a basso livello e, soprattutto, è soggetto a differenti interpretazioni. Ne consegue che nonostante l'interoperabilità tra le smart card sia stata, ad un certo livello, effettivamente raggiunta, la programmazione delle smart card è rimasta ancora un'attività ristretta ad una grande quantità di specialisti del settore, tipicamente molto costosi, e comunque dipendente dal terminale di lettura. In tale panorama, numerose aziende del settore hanno formato dei gruppi di lavoro per tentare di definire un documento di specifiche che potessero rendere finalmente interoperabili smart card e lettori di diversa fattura, e consentissero di semplificare lo sviluppo di applicazioni con smart card, avvicinandolo il più possibile ai paradigmi di programmazione tradizionali.

In questo modo sono nati:

- **il PC/SC Workgroup** che ha definito le specifiche PCISCS le quali, tra le altre cose, hanno il pregio di aver standardizzato i terminali di lettura su piattaforma Microsoft mediante una definizione precisa dei requisiti hardware e delle interfacce tra terminale, PC e Sistema Operativo;
- **le specifiche PKCS#11 e PKCS#15** che propongono un modello di dispositivo crittografico astratto che abbraccia tutte le smart card crittografiche, definiscono una API ad alto livello in C/C++ e Java per la scrittura di applicazioni portabili che interagiscono con la smart card e infine stabiliscono la struttura logica e fisica del file system sulla smart card;

- **OpenCard Consortium** che ha definito l'OpenCard Framework costituito da un insieme di classi Java il cui obiettivo è proporre da un lato un'interfaccia unica con la smart card e con il lettore al fine di assicurare l'interoperabilità e dall'altro un modello di programmazione basato su un linguaggio di programmazione semplice, diffuso e multiplatforma qual'è Java;
- **Le specifiche EMV** che definiscono le caratteristiche di una smart card che deve operare in ambito bancario.

Sun Microsystems invece, seguendo una filosofia diversa, ha proposto una nuova tecnologia basata sul linguaggio Java, denominata javaCard, che ridisegna completamente il Sistema Operativo della smart card installando al suo interno la **virtual machine** Java.

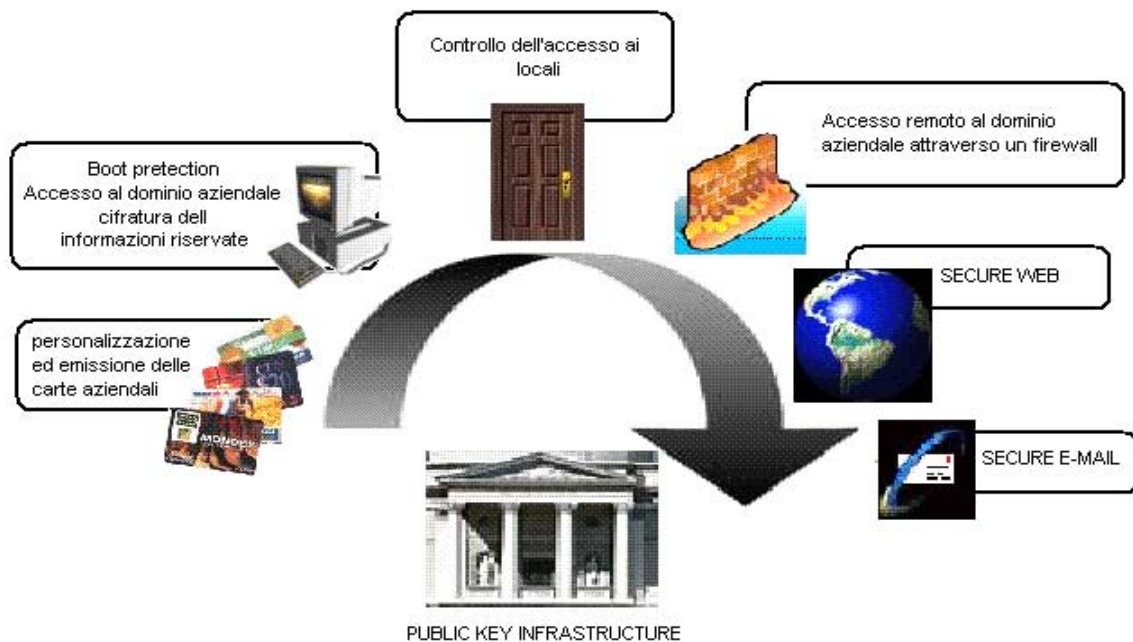
Il risultato è una smart card le cui funzionalità possono essere facilmente estese e personalizzate usando il linguaggio Java con una considerevole diminuzione dei tempi e dei costi di sviluppo.

1.7 CORPORATE CARD

La **Corporate Card** è una carta aziendale multifunzionale rilasciata da una azienda ai propri dipendenti in sostituzione del classico badge per fornire, oltre alle funzioni classiche di un badge (controllo dell'accesso ai locali, marcatura dei cartellini, conteggio dei pasti in mensa ecc.), anche dei servizi avanzati che vanno dalla protezione della postazione di lavoro attraverso un sistema di controllo dell'accesso, alla navigazione sicura sul Web tramite certificati digitali, alla protezione della posta elettronica mediante cifratura e firma digitale delle email, per finire con l'accesso remoto al dominio aziendale controllato dal certificato installato sulla Corporate Card (utile soprattutto nel telelavoro).

L'immagine che segue mostra i principali servizi che tipicamente vengono offerti ai dipendenti tramite la Corporate Card.

Sistemi informatici di questo tipo si basano solitamente su di una Public Key Infrastructure (PKI) che è sostanzialmente un sistema di supporto alla crittografia a chiave pubblica. In tali sistemi la smart card assume un ruolo determinante poiché fornisce uno strumento adeguato a proteggere le chiavi private.



1.7.1 Carta Multiservizi della Difesa

La Carta Multiservizi della Difesa (CMD) è una smart card di identificazione del militare che contiene i dati personali, la foto, l'impronta digitale, i dati sanitari e i certificati digitali necessari all'identificazione e alla firma elettronica. Il progetto ideato e definito nei suoi requisiti essenziali in ambito Amministrazione Difesa è affidato a Siemens Informatica che, oltre alla fornitura delle carte, ha la responsabilità della progettazione, dell'installazione e della configurazione dell'infrastruttura a chiave pubblica (PKI) per il rilascio dei certificati di autenticazione e di firma digitale.

Grazie all'inserimento della CMD in un Pocket PC, è possibile verificare l'autenticità della smart card e l'autenticità dei dati personali ivi registrati. Inoltre, nella CMD possono essere inseriti, anche successivamente all'emissione, altri servizi tipici delle Forze Armate quali, ad esempio, la gestione del vestiario e degli stipendi.

Le principali caratteristiche sono:

- **Documento di riconoscimento:** permette di operare l'identificazione del possessore a vista, mediante i dati stampati sul supporto plastico, elettronicamente, leggendo i dati contenuti nel microchip (in questo caso è possibile verificarne l'autenticità), o in rete, mediante il certificato digitale (chiave RSA). La CMD garantisce la piena e completa interoperabilità con la Carta d'Identità Elettronica (CIE) e la Carta Nazionale Servizi (CNS) che nel prossimo futuro saranno utilizzate (da tutti i cittadini). Le informazioni riportate sulla carta sono infine completate con il codice "International Civil Aviation Organization" (ICAO) e con le informazioni previste per i prigionieri di guerra;

- **Impronte digitali:** all'interno del microchip sono contenute le impronte digitali del titolare, pertanto la CMD può essere utilizzata per effettuare il riconoscimento biometrico;
- **Dati personali:** sul supporto plastico della CNID e nel microchip sono presenti i dati del titolare. In particolare le informazioni di natura privata (ad es. dati sul trattamento economico) sono contenute esclusivamente nel microchip e sono accessibili solo digitando un PIN;
- **Dati sanitari militari:** nel microchip sono contenuti i dati sanitari del titolare secondo le specifiche espresse dallo Stato Maggiore della Difesa. E' assicurata la piena compatibilità in materia sanitaria con la Carta d'Identità Elettronica tramite l'utilizzazione della stessa struttura ("NetLink" - protocollo di standardizzazione dei dati sanitari a livello internazionale). I dati sanitari d'interesse dell'Esercito sono protetti da PIN, mentre i dati sanitari di emergenza sono di libera lettura;
- **Certificati Digitali:** è possibile memorizzare ed utilizzare fino a 4 certificati digitali per poter installare servizi aggiuntivi.

La CMD si basa su una Smart Card di tipo multiapplicativo che possiede le capacità crittografiche richieste per l'esecuzione delle operazioni di firma digitale e cifratura dei dati. La smart card dispone di una memoria per le applicazioni pari a 32 KB, mentre il chip ed il sistema operativo sono certificati ITSEC e4-high (che corrisponde al livello di sicurezza più elevato richiesto in Europa per le Smart Card crittografiche).

Le applicazioni di prevista utilizzazione sono: firma digitale, cifratura/decifratura dati, autenticazione ed accesso sicuro (cifrato) al Sistema Informativo di Forza Armata (ad es. l'Esercito consente l'accesso al portale del Sistema Informativo Gestionale dell'Esercito -SIGE - garantendo l'identificazione forte di colui che accede e proteggendo i dati trasmessi tramite cifratura), posta sicura mediante e-mail firmate e/o cifrate con garanzia di autenticità del mittente, integrità e sicurezza, accesso in totale sicurezza alla postazione di lavoro (eventualmente congiunto all'identificazione dell'impronta digitale), ingresso ad aree riservate sottoposto ad autenticazione forte tramite utilizzazione congiunta della CMD e delle impronte digitali in essa contenute, accesso a servizi di prelievo armi e carburanti, "passi" per ingresso in specifici locali (mense, uffici, ecc ...), acquisto beni presso strutture militari in operazioni, ecc.

Fondamenti tecnologici

Una teoria scientifica è solo un modello dell'universo, o di una sua parte, limitata, è un insieme di regole che mettono in relazione le quantità presenti nel modello con le osservazioni che facciamo nella realtà. Il modello esiste solo nella nostra mente e non ha alcun'altra realtà.

Stephien I lawking

Il termine smart card sottintende un insieme di tecnologie, comprendenti circuiti integrati, microprocessori, memorie RAM, ROM EEPROM, antenne, ecc., integrate nello stesso circuito elettrico per formare un **microchip** che è il "cuore" della smart card. Questo capitolo presenta una panoramica su tali tecnologie e sulle modalità di integrazione descrivendo in particolare le varie tipologie di smart card presenti sul mercato.

2.1 LA SMART CARD

La smart card è un dispositivo hardware delle dimensioni di una carta di credito che possiede potenzialità di elaborazione e memorizzazione dati ad alta sicurezza (come mostrato nell'immagine sotto riportata). E' costituita da un supporto di plastica nel quale è incastonato un **microchip** connesso ad un'interfaccia di collegamento che può essere una contattiera o **un'antenna**. Il microchip fornisce funzionalità di calcolo e memorizzazione dati; la contattiera o l'antenna consentono al microchip di dialogare con uno speciale terminale di lettura collegato solitamente ad un computer mediante porta seriale, parallela, USB, ecc.



Le smart card possono essere classificate in base a diversi criteri:

- sulla base delle potenzialità e delle caratteristiche del microchip, si distinguono smart card a sola memoria (**Memory Card**) e smart card a microprocessore (**Microprocessor Card**);
- sulla base del tipo di interfaccia di collegamento, si distinguono smart card con contattiera (**Contact**), smart card con antenna (**Contactless**) e smart card con antenna e contattiera (**Dual Interface**).

Le caratteristiche del microchip determinano sia il costo della smart card, sia l'ambito di applicazione. Come si vedrà nei prossimi paragrafi, le smart card a sola memoria, tecnologicamente più semplici, sono più economiche ma offrono un livello di sicurezza più basso rispetto alle smart card a microprocessore e pertanto sono usate tipicamente per conservare dati che non necessitano di protezione "forte".

Le caratteristiche del supporto di plastica e dell'interfaccia di comunicazione determinano invece il ciclo di vita della smart card.

2.2 MEMORY CARD

La **memory card**, o smart card a sola memoria, offre unicamente funzionalità di memorizzazione sicura dei dati.

Il microchip contiene una componente di memoria permanente nella quale si può leggere e scrivere attraverso un insieme di funzioni cablate in un circuito logico preprogrammato, stampato nel microchip durante la fase di produzione.

Il circuito logico a sua volta comprende un meccanismo di protezione che salvaguarda l'accesso ai dati memorizzati (basato tipicamente su un insieme di permessi di accesso). Tipicamente le **memory card** offrono circa da 1 a 4 Kbyte di memoria e sono usate principalmente per applicazioni semplici quali carte prepagate, carte per la raccolta punti, ecc. In questi casi, il meccanismo di protezione consente di evitare l'incremento fraudolento del credito.

Il vantaggio delle **memory card** sta nel loro basso costo dovuto alla semplicità della tecnologia adottata, tuttavia per applicazioni più complesse che richiedono un livello di sicurezza maggiore, si preferisce usare smart card a microprocessore.

2.3 MICROPROCESSOR CARD

La smart card a microprocessore, grazie alla potenza di calcolo fornita da un microprocessore incluso nel microchip, può essere paragonata ad un piccolo computer portatile altamente affidabile e inattaccabile in grado di elaborare e memorizzare informazioni salvaguardandone la riservatezza.

Nella memoria del microchip è installato un sistema operativo che implementa la logica operativa della smart card. In particolare il sistema operativo si occupa della gestione interna della memoria e fornisce funzioni di lettura e scrittura in memoria, funzioni di programmazione dei permessi di accesso, funzioni crittografiche, ecc.

L'accesso alla memoria è subordinato ad un insieme di permessi di accesso, programmabili in maniera puntuale mediante le funzioni del sistema operativo, che sono concessi in base all'esito di specifici test quali verifica di PIN, password, firme digitali, informazioni biometriche, ecc.

La programmabilità dei microchip, conseguente alla presenza di un sistema operativo, consente di ottimizzare e personalizzare la smart card per una particolare applicazione o di integrare sullo stesso dispositivo più applicazioni (eventualmente interagenti tra loro). L'unico limite a tale flessibilità è rappresentato dalla disponibilità di risorse di memoria. Attualmente le smart card a microprocessore sono ampiamente usate in sistemi di controllo degli accessi basati su procedure di **Strong Authentication** in applicazioni bancarie per validare le transazioni mediante firma digitale, in sistemi wireless (GSM, UMTS, ecc), e in tutti i sistemi dove è richiesto un alto livello di sicurezza e di privacy (Carta d'Identità Elettronica, Voto Elettronico, Badge Aziendale, ecc.).

2.4 SMART CARD CONTACT, CONTACTLESS E DUAL INTERFACE

La differenza tra le smart card **contact** e **contactless** è nel tipo di interfaccia di collegamento tra il microchip e il mondo esterno. Le prime hanno una contattiera (Figura A) mediante la quale ricevono l'alimentazione e dialogano con l'esterno una volta inserite in un apposito lettore di smart card (Figura B). Le seconde hanno un'antenna (Figura C), che reagisce alla presenza di un campo elettromagnetico emesso da uno speciale dispositivo di lettura/scrittura nella banda delle radio-frequenze, consentendo al microchip di scambiare dati con l'esterno (purché l'antenna si trovi entro una distanza minima dal dispositivo di lettura/scrittura). Poiché le smart card **contact** devono necessariamente essere inserite (nel verso giusto) in un apposito lettore di smart card, le **contactless** sono preferite in contesti in cui sono richieste transazioni veloci, come sistemi di controllo dell'accesso ad edifici e aree riservate o sistemi di **ticketing** nel trasporto pubblico.

Le smart card **contactless**, inoltre, risultano meno soggette ad usura rispetto alle smart card **contact** poiché, in queste ultime la contattiera è sottoposta a sfregamento a causa dei ripetuti inserimenti nel lettore. Le **contactless** tuttavia non consentono di eseguire transazioni lunghe e complesse poiché la comunicazione **wireless** che si instaura con il dispositivo di lettura deve avvenire necessariamente entro una distanza minima.

Più specificamente, la smart card **contactless** si sposta molto velocemente in prossimità del dispositivo di lettura, seguendo il passo dell'individuo che la porta in tasca o nel portafogli, e pertanto, potrebbe uscire dalla zona di influenza del campo elettromagnetico causando l'interruzione della comunicazione. In secondo luogo, i protocolli di comunicazione **wireless** adottati non consentono di accedere a tutte le funzionalità offerte dal sistema operativo ma, tipicamente, permettono di eseguire solo semplici operazioni di lettura/ scrittura in memoria. Per tali ragioni, l'interfaccia **contactless** è tipicamente usata per nelle smart card a sola memoria, mentre l'interfaccia **contact** viene montata su smart card a microprocessore usate per realizzare applicazioni che richiedono interazioni lunghe e complesse.

Le smart card **dual-interface**, offrono entrambe le interfacce **contact** e **contactless** e pertanto la comunicazione con il microchip può avvenire indifferentemente mediante una o l'altra. Tale caratteristica consente di integrare sulla stessa smart card sia applicazioni complesse come quelle di firma digitale tipiche delle smart card **contact**, sia applicazioni più semplici e veloci, come quelle di controllo dell'accesso ad aree riservate, che richiedono esclusivamente accessi alla memoria **wireless**.

Esiste, infine, un ultimo tipo di smart card a tecnologia **ibrida** che include, in maniera indipendente, entrambe le tecnologie **contact** e **contactless**. In pratica nello stesso supporto di plastica sono incastonati due microchip uno collegato ad una contattiera ed uno connesso ad una antenna. Tali microchip non comunicano tra loro direttamente ma possono comunicare con l'esterno in maniera indipendente attraverso le due interfacce. Sebbene offrano diverse caratteristiche interessanti, non sono molto diffuse poiché richiedono un processo di produzione particolarmente complesso nel quale due microchip, una contattiera ed un'antenna devono essere incastonati nello stesso supporto di plastica.



A



B

C

2.1.4 Token USB

In alternativa al supporto di plastica in cui è incastonato il microchip, nonché alla contattiera o all'antenna, alcuni produttori hanno proposto un dispositivo basato su tecnologia USB. Più specificamente, lo stesso microchip che tipicamente viene installato sulla smart card, viene in questo caso microsaldato ad un connettore USB (completo di firmware) e il modulo risultante viene inserito in un piccolo astuccio simile ad un portachiavi (come mostrato in figura).

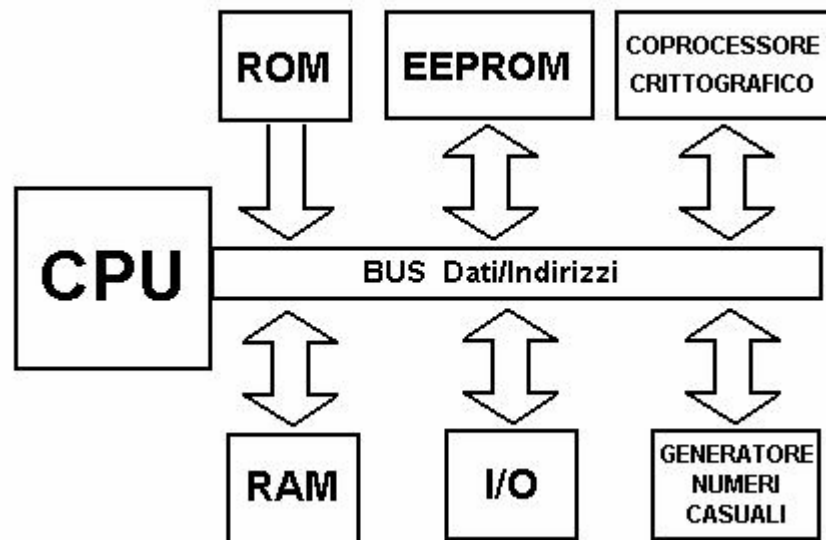


Il dispositivo, solitamente chiamato **Token USB**, offre le stesse potenzialità di una smart card a microprocessore senza richiedere la presenza di un lettore di smart card connesso al computer.

Vista la sostanziale equivalenza funzionale tra la smart card classica e il **Token USB**, i temi trattati nelle prossime parti si riferiscono indifferentemente a entrambi i dispositivi.

2.2 Architettura del microchip

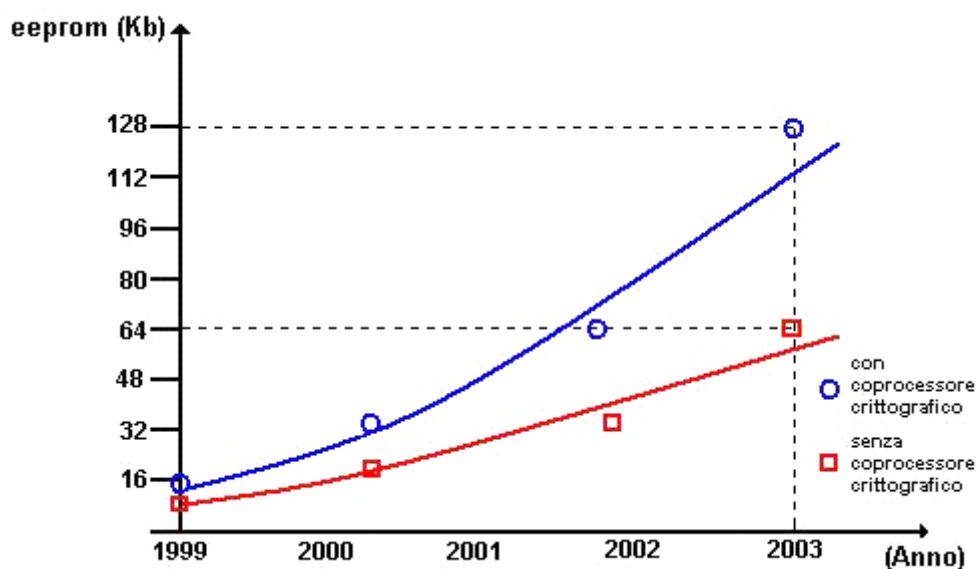
Il microchip è l'unità che conferisce alla smart card potenzialità di calcolo e memorizzazione dati (la figura sotto riportata ne mostra l'architettura generale).



Si individuano diverse componenti:

- un **microprocessore** (CPU) che conferisce capacità computazionali. Nelle smart card attualmente in commercio sono usati principalmente microprocessori a 8 bit con una frequenza di clock pari a 5 MHz con il set di istruzioni derivante dal processore Motorola 6805 o Intel 8051 e con bus di indirizzamento a 16 bit. Si tratta in pratica di CPU paragonabili a quelle installate nei personal computer dei primi anni '80 (Commodore 64, Sinclair ZX Spectrum, PC IBM, ecc.). Alcuni microprocessori ad alte prestazioni includono un moltiplicatore di frequenza che consente di arrivare fino a 40 MHz; tuttavia, la nuova generazione di smart card è dotata di microprocessori a 16 o 32 bit con un set di istruzioni ridotto di tipo RISC;
- una **RAM** (*Random Access Memory*) usata dal microprocessore per conservare i risultati parziali della computazione. Tipicamente contiene da 100 a 2K byte. La RAM è una memoria non persistente che viene cancellata appena il microchip non è più alimentato;
- una **ROM** (*Read Only Memory*), in cui sono conservati i dati "statici", ossia non modificabili dopo la produzione del microchip, quali il sistema operativo e i dati relativi al microchip (serial number, produttore, ecc.). Generalmente le sue dimensioni variano da 2 a 96 Kbyte. La ROM è una memoria persistente che conserva i dati anche quando il microchip non è più alimentato.

- una **EEPROM** (*Electrical Erasable Programmable Read Only Memory*), che può essere paragonata ad un hard disk di un comune PC. poiché consente di leggere e scrivere dati che persistono anche quando viene tolta l'alimentazione nella smart card a microprocessore attualmente in commercio le dimensioni della EEPROM sono tipicamente 16, 32 o, nelle più avanzate, 64 KB. Tuttavia, come mostrato nel grafico sottostante le capacità di memoria della EEPROM crescono seguendo l'evoluzione della tecnologia a semiconduttore. La EEPROM, per le sue peculiarità elettriche, presenta alcuni limiti. In primo luogo, il numero di cicli di scrittura (*Write Cycles*) e il tempo di conservazione dei dati (*Retention Time*) sono limitati. In secondo luogo, il tempo richiesto per la scrittura dei dati è maggiore rispetto a quello previsto per la RAM. Nella maggior parte delle smart card la EEPROM consente circa 100,000 cicli di Scrittura, ha un tempo di conservazione dei dati pari a 10 anni e in scrittura è circa 1000 volte più lenta della RAM. Attualmente sono in fase di Studio e test microchip dotati di memoria Flash in sostituzione della EEPROM.



- una **porta di I/O** seriale (ad un solo bit) mediante la quale il microchip comunica con l'esterno. Le smart card *contact* supportano tipicamente velocità dell'ordine di 9600 bps, mentre le smart card *contactless* sono solitamente più veloci.
- un **coprocessore crittografico**, che consente di accelerare le operazioni in modulo su numeri interi molto grandi. Tali operazioni sono largamente usate negli algoritmi crittografici a chiave pubblica;
- un **generatore di numeri casuali** (presente solo nei microchip più evoluti) usato solitamente nella generazione delle chiavi crittografiche e negli algoritmi di autenticazione di tipo *challenge-response*,
- un **bus dati** e un **bus indirizzi** che mettono in comunicazione le componenti dei microchip.

2.6 PRODUZIONE DELLE SMART CARD

Tipicamente il ciclo di produzione delle smart card è suddiviso in quattro fasi:

- **produzione del chip**. il microchip viene prodotto a partire da un wafer di silicio adottando le usuali tecniche di produzioni di componenti a semiconduttore. In questa fase il nucleo del sistema operativo e le altre informazioni statiche sono scritte definitivamente nella ROM mediante l'operazione di **mascheratura**. Più specificamente il produttore prepara una maschera della ROM (contenente tutti i dati che devono essere scritti nella stessa) che viene poi stampata nel wafer di silicio;
- **installazione**. Il microchip, chiamato in questa fase **micromodulo**, viene saldato alla contattiera mediante tecniche avanzate di micro-elettronica
- **embedding**. il modulo composto da microchip e contattiera viene incastonato nel supporto plastico delle dimensioni di una carta di credito;
- **inizializzazione**. Il microchip viene inizializzato elettricamente, pronto per essere programmato.

scomposizione di una smart card evidenziando le componenti in gioco nelle fasi descritte.



2.7 IL SISTEMA OPERATIVO

La ROM ospita il sistema operativo che implementa la logica operativa della smart card. L'obiettivo primario di un sistema operativo è gestire in maniera sicura, flessibile ed efficiente la memoria disponibile nella EEPROM. Come qualsiasi sistema operativo, è composto da una serie di moduli che gestiscono la memoria e le risorse del microprocessore e implementano l'I/O. In particolare gestisce a basso livello le operazioni di lettura, scrittura e ottimizzazione della memoria (allocazione e rilascio di memoria, deframmentazione, ecc.), esegue le operazioni di I/O attraverso l'interfaccia contact o contacless, gestisce i permessi di accesso alla memoria e fornisce una insieme di funzioni, atte a realizzare veri e propri programmi memorizzati nella EEPROM.

L'accesso alle risorse della smart card da parte di un applicazione esterna è gestito dal livello più alto del sistema operativo che implementa un framework, definito dallo standard ISO 7816 (che sarà trattato in seguito), comprendente funzioni di lettura e scrittura dei dati, algoritmi di verifica del PIN, funzioni di generazione di chiavi crittografiche, di firma digitale, ecc.

2.8 PROTEZIONE DEI DATI SULLA SMART CARD

Il microchip offre due distinti livelli di protezione dei dati conservati nella smart card; un livello fisico ed un livello logico. La protezione a livello fisico è gestita dal produttore del microchip che provvede a proteggere la EEPROM con un meccanismo che renda impossibile (o almeno estremamente difficile e costoso) leggere il contenuto della memoria e di conseguenza eviti la clonazione dei microchip. Tale protezione, almeno in linea di principio, potrebbe essere violata con attacchi diretti al substrato di silicio che compone il microchip. Ad esempio, considerando che le informazioni riservate sono memorizzate nella EEPROM e che le relative operazioni di scrittura dipendono fortemente dalle tensioni applicate alle giunzioni elettriche e dalla temperatura di lavoro, sottoponendo il microchip a determinati livelli di tensione (o più specificamente a particolari forme d'onda) e a particolari valori di temperatura, si potrebbero leggere le informazioni memorizzate.

In teoria, si potrebbe anche agire direttamente sullo strato di semiconduttore che compone il microchip, cercando di analizzare al microscopio le giunzioni elettriche in modo da leggere lo stato dei singoli bit della memoria.

Per proteggere le smart card da tali tipologie di attacco, i produttori di microchip hanno predisposto dei sistemi di allarme che intervengono in caso di violazione, cancellando immediatamente il contenuto della EEPROM e bloccando il microprocessore, rendendo così la smart card inutilizzabile.

Il livello logico è invece gestito dall'entità che inizializza e personalizza la smart card (entità emittente) mediante le funzionalità offerte dal sistema operativo.

I dati inseriti nel microchip sono protetti in lettura/scrittura mediante un PIN (***Personal Identification Number***) di cui solo il titolare (ossia la persona a cui viene data la smart card) è a conoscenza.

Il sistema operativo consente l'accesso alle informazioni protette solo dopo aver verificato che il PIN ricevuto corrisponde a quello corretto. In particolare, se qualcuno tentasse di accedere alla carta inserendo ripetutamente PIN diversi nella speranza di individuare quello giusto (attacco di forza bruta), dopo un certo numero di tentativi falliti il sistema operativo reagirebbe all'attacco bloccando il PIN e negando di conseguenza l'accesso alla memoria fino ad un eventuale successivo sblocco mediante il **PUK** (**PIN Unblocking Key**).

Le smart card corredate di co-processore crittografico forniscono un metodo alternativo (oppure da usare in congiunzione) al PIN per ottenere i permessi di accesso alla memoria. Esso si basa su un algoritmo di firma digitale del tipo **challenge-response**: l'applicazione che vuole accedere alla smart card chiama una particolare funzione del sistema operativo; questo, in risposta, genera una stringa casuale di bit su cui l'applicazione appone una firma digitale (mediante la propria chiave privata) restituendo la stringa firmata alla smart card; il sistema operativo, usando una copia della chiave pubblica (corrispondente alla suddetta chiave privata), precedentemente memorizzata sulla smart card, verifica la firma digitale e, in caso di successo, concede i permessi di accesso.

2.9 QUALE SMART CARD?

La scelta di una particolare tipologia di smart card per l'implementazione di un'applicazione (o insieme di applicazioni) ha un notevole impatto sia sul costo dell'intera soluzione sia sull'architettura del sistema. I principali parametri da prendere in considerazione sono:

- quantità di dati da memorizzare sulla smart card;
- tipo di comunicazione **contact o contactless**;
- livello di riservatezza dei dati memorizzati.

Se le informazioni da memorizzare sulla smart card sono poco voluminose e non richiedono un alto livello di protezione possono essere memorizzate su smart card a sola memoria **contact o contactless**.

Se invece hanno consistenza maggiore e richiedono un alto livello di protezione occorre usare smart card a microprocessore **contact** dotate di coprocessore e funzionalità crittografiche.

Lo standard ISO 7816

L'aspetto divertente degli standard è che ce ne sono così tanti tra cui scegliere
Andrew Tanenbaum

L'uso della smart card richiede necessariamente un sistema di elaborazione composto da un computer al quale è collegato un dispositivo terminale d'accesso alla smart card mediante porta seriale, parallela, USB, ecc. La varietà e l'eterogeneità dei dispositivi terminali, nonché delle stesse smart card ha portato alla definizione di un insieme di specifiche, diventate poi uno standard ISO, che stabiliscono le caratteristiche fisiche, elettriche e operative che consentono l'interoperabilità tra smart card e terminali di lettura provenienti da produttori diversi. Tali specifiche sono definite nello standard ISO 7816.

3.1 LE SPECIFICHE TECNICHE STANDARD

Lo standard internazionale ISO 7816, denominato "*Identification Cards - Integrated circuit(s) cards with contact*", definisce le caratteristiche fisiche, elettriche e operative delle smart card a microprocessore con contatti elettrici (*contact*). Lo standard consiste delle seguenti dieci parti:

- **Parte 1: Caratteristiche Fisiche**
Descrive le caratteristiche fisiche che la smart card deve avere al termine della fase di produzione: resistenza a torsione, risposta a campi elettromagnetici e a cariche elettriche statiche, ecc.. Definisce inoltre la temperatura di lavoro;
- **Parte 2: Dimensioni e posizione dei contatti elettrici**
Specifica le dimensioni, la posizione relativa al supporto di plastica e l'assegnazione di ciascuno dei contatti elettrici che compongono la contattiera;
- **Parte 3: Segnali elettrici e protocolli di trasmissione**
Descrive i valori di tensione e di corrente dei segnali elettrici per ciascuno dei contatti e i meccanismi di trasmissione tra la smart card e il dispositivo d'interfacciamento (un terminale di lettura/scrittura detto tipicamente lettore di smart card);
- **Parte 4: Comandi per l'intercambiabilità**
Descrive i comandi, i messaggi e le risposte trasmessi tra la smart card e il terminale d'interfacciamento, la struttura logica dei file e dei dati memorizzati nella memoria della smart card nonché i meccanismi di protezione dell'accesso agli stessi. Definisce inoltre i comandi crittografici per cifrare la comunicazione tra la smart card e il terminale (*Secure Messaging*);

- **Parte 5: *Procedure di registrazione e sistemi di numerazione per identificativi d'applicazione***
 Descrive le regole per registrare e identificare univocamente le applicazioni installate sulla smart card;
- **Parte 6: *Definizione dei dati***
 Definisce il formato e le regole di costruzione dei dati memorizzati sulla smart card e scambiati con il terminale;
- **Parte 7: *Comandi per query strutturate***
 Definisce i concetti di un database su smart card con relativo **SCQL (Structured Card Query Language)** e i relativi comandi;
- **Parte 8: *Comandi relativi alla sicurezza***
 Specifica i protocolli di sicurezza da usare con la smart card, i comandi atti ad eseguire algoritmi crittografici sia in chiave privata sia in chiave pubblica e a maneggiare i relativi certificati, nonché ulteriori comandi per la trasmissione sicura dei dati;
- **Parte 9: *Comandi aggiuntivi e attributi di sicurezza***
 Descrive il ciclo di vita di una smart card con i relativi attributi di sicurezza e ulteriori comandi di gestione;
- **Parte 10: *Segnali elettrici e 'Answer to Reset'***
 Definisce i segnali elettrici e le modalità di costruzione della stringa "**Answer to Reset**" in risposta al comando RESET.

Le prime tre parti sono seguite dalla totalità dei produttori di smart card poiché definiscono le caratteristiche fisiche ed elettriche della smart card e in particolare del microchip. Le altre, che descrivono principalmente le strutture dati, i comandi e i protocolli di comunicazione, sono viste solitamente come indicazioni generali e spesso molti costruttori adottano soluzioni parziali e/o alternative.

3.2 CARATTERISTICHE FISICHE ED ELETTRICHE

La parte 1 definisce le seguenti caratteristiche elettrico-fisiche:

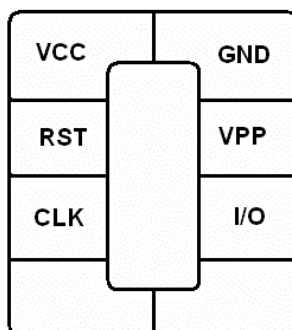
- dimensioni del supporto di plastica: mm 85,6 x 53,97 x 0,76 (le dimensioni di una carta di credito)
- temperatura di lavoro: compresa tra 0° C e 50° C;
- resistenza meccanica: la contattiera deve rimanere inalterata se sottoposta ad una pressione pari a quella esercitata da una sfera d'acciaio di diametro 1 mm alla quale è applicata una forza di 1.5 Newton;
- resistenza elettrica: la resistenza misurata tra due qualsiasi contatti deve essere minore di 0.5 Ohm se sottoposta ad una corrente tra 50 microampere e 300 microampere;

- resistenza a radiazioni X: la carta non deve mostrare malfunzionamenti se esposta da entrambi i lati a una dose pari a 0.1 Gy di radiazione X con energia media compresa tra 70 e 140 KeV.

La parte 2 definisce le dimensioni la posizione e la funzione degli 8 contatti che compongono la contattiera come mostrato in **Figura 3.1**. La **Tabella 3.1** mostra la funzione assegnata a ciascun contatto.

La parte 3 specifica il comportamento e i valori di tensione e di corrente per ciascun pin della contattiera nonché il protocollo di risposta al segnale di RESET, detto ATR (Answer to Reset), che consente di identificare il tipo di microchip e il protocollo di trasmissione (come si vedrà in seguito). In particolare, la parte 3 definisce le seguenti caratteristiche

Disposizione dei contatti e segnali presenti sulla contattiera



Contatto	Descrizione
VCC	Alimentazione
RST	Segnale di reset
CLK	Segnale di clock
GND	Massa
VPP	Alimentazione aggiuntiva (Programming voltage)
I/O	Canale di I/O

- **Vcc**: il contatto è usato per fornire al microchip l'alimentazione necessaria al suo funzionamento. I valori ammessi sono: 5 V o 3 V e definiscono rispettivamente smart card di classe A o classe B;
- **I/O**: il contatto mediante il quale avviene la trasmissione seriale dei dati dalla smart card al dispositivo e viceversa;

- **CLK**: il contatto riceve il segnale di clock destinato al microchip. I valori ammessi sono: da 1 a 5 MHz per la classe A o da 1 a 4 MHz per la classe B.
- **RST**: il contatto riceve il segnale di RESET;
- **Vpp**: per la classe A il contatto può essere usato per fornire al microchip l'alimentazione aggiuntiva richiesta per scrivere o cancellare (o in generale programmare) dati nella EEPROM.

3.3 L'ATR

L'ATR, descritto nella parte 3, è per definizione il valore della sequenza di byte inviata dal microchip in risposta ad un segnale di RESET. In pratica, ogni operazione di RESET induce una risposta da parte del microchip sul contatto di I/O formata al massimo da 33 caratteri che codificano una serie di informazioni relative alle caratteristiche del microchip, ai protocolli di trasmissione supportati e al sistema operativo installato. I 33 caratteri dell'ATR possono essere suddivisi in cinque sezioni:

- 1) un carattere iniziale TS
- 2) un carattere di formato TO
- 3) diversi caratteri di interfaccia TA_n , TB_n , TC_n , TD_n
- 4) al massimo 15 "Historical Characters"
- 5) un carattere di checksum TCK

TS definisce la convenzione per codificare i byte seguenti. Le specifiche definiscono due convenzioni: diretta, caratterizzata dal valore esadecimale 3B, inversa, associata al valore 3F.

TO è il carattere di formato. Il primo nibble indica i seguenti caratteri di interfaccia. Il secondo nibble indica il numero di historical characters.

TA_n , TB_n , TC_n , TD_n specificano i dettagli dell'interfaccia con il lettore quali il protocollo di comunicazione T=0 o T=1, la frequenza di clock, il voltaggio il baud rate, ecc.

Gli Historical Characters contengono informazioni di vario tipo quali, ad esempio, il produttore della smart card, la versione del sistema operativo installato, ecc.

TCK, infine, è il carattere di checksum definito come XOR di tutti i byte dal TO al TCK.

3.4 PROTOCOLLI DI TRASMISSIONE

La parte 3 definisce anche i protocolli di trasmissione tra microchip e terminale: protocollo T = 0 e protocollo T = 1.

3.4.1 Protocollo T = 0

T = 0 è un protocollo orientato al byte (nel senso che la quantità minima di informazione scambiata è un singolo byte) che implementa una "*half-duplex transmission of asynchronous characters*". Per ogni byte ricevuto, il microchip esegue un controllo di parità e invia un bit di risposta (ACK, acknowledge) impostato a 0 o a 1 per avvisare il trasmettitore dell'avvenuta ricezione. L'inconveniente di tale protocollo è che il microchip tipicamente deve memorizzare in RAM o in EEPROM l'intera sequenza di byte ricevuti prima di poterla interpretare ed eseguire le dovute operazioni e pertanto, le trasmissioni con T = 0 risultano talvolta lente.

Il protocollo T = 0 è usato principalmente dai microchip destinati alle SIM card in ambito GSM

3.4.2 Protocollo T = 1

T = 1 è un protocollo orientato ai blocchi di byte che implementa una "*half-duplex asynchronous transmission of blocks*". È un protocollo più sofisticato e veloce del T = 0 poiché la trasmissione dei dati avviene in blocchi di byte detti *I-block*. Il controllo di correttezza viene eseguito sul singolo *I-block* mediante un byte detto *Longitudinal Redundancy Check* calcolato con un semplice XOR dei dati che compongono il blocco.

Il protocollo T = 1 prevede inoltre un segnale speciale detto *WTX (Waiting Time Extension)* che consente al microchip di richiedere al trasmettitore più tempo per terminare l'elaborazione del comando appena ricevuto.

T = 1 è usato tipicamente (la tutte le smart card multiapplicazione di ultima generazione che non sono destinate al mercato GSM).

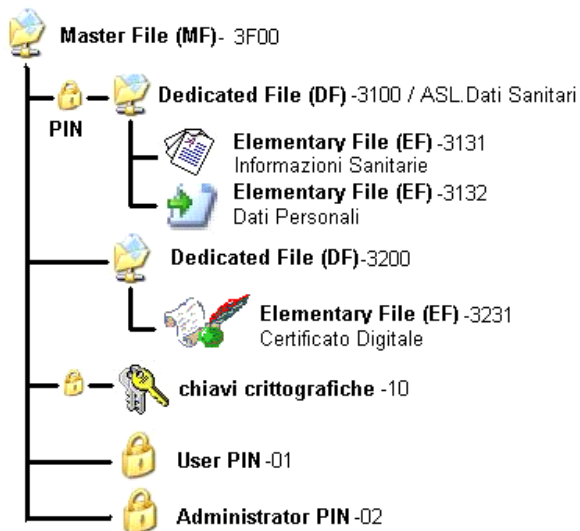
3.5 STRUTTURA E FORMATO DEI DATI MEMORIZZATI NELLA EEPROM

La parte 4 definisce la struttura logica e il formato dei dati che possono essere memorizzati nella EEPROM, nonché i comandi che devono essere implementati dal sistema operativo.

3.5.1 File System

Dal punto di vista logico, la EEPROM è organizzata in un file system gerarchico composto da file e directory del tutto simile a quello di un comune Hard Disk (come mostrato in figura). La radice, detta **Master File** (MF), contiene una serie di sottodirectory, chiamate **Dedicated File** (DF).

Organizzazione logica gerarchica dei file system di una EEPROM



Ogni **Dedicated File** (così come **il Master File**) può ospitare dei **Dedicated File**, dei file denominati **Elementary File** (EF) o degli oggetti speciali detti **Security Data Object** (SDO) e **Security Environment Object** (SEO).

Gli Elementary File Ospitano i dati mentre i **Dedicated File** separano logicamente gli **Elementary File**. Con tale strutture è possibile raggruppare in **Dedicated File** "dedicati (da cui il nome **Dedicated File**) visibile nella figura sotto riportata riservati alle diverse applicazioni che agiscono sulla smart card.

Gli Elementary File e i **Dedicated** visibile nella figura sotto riportata possono essere identificati mediante:

- un "nome" detto FILE ID (FID) formato da due caratteri esadecimale (ad esempio la parte 4 specifica che il FID del **Master File** deve essere "3F00");
- un "path name" ottenuto concatenando il **File ID** dei **Dedicated File** che precedono **l'Elementary File o il Dedicated File** (ad esempio il file "dati personali" visibile nella figura sotto riportata è identificato dal **path** 3F00 3100 3131);
- un nome "corto" detto **SHORT ID** (**SID**) che permette di identificare velocemente un file mediante un numero compreso tra 1 e 32 (e quindi con soli 5 bit).

In aggiunta alle tre regole appena descritte, i Dedicated File possono essere identificati anche con una stringa lunga al massimo 16 byte detta **Application identifier** (AID Ad esempio il DF 3 100 può essere identificato anche con l'AID "**ASL.DatiSanitari**").

I **Security Data Object**, identificati da un **Security Data Object** ID, consentono di memorizzare oggetti relativi alla sicurezza quali PIN, chiavi crittografiche, informazioni biometriche, ecc. mediante i quali è possibile assegnare i permessi di esecuzione di specifiche azioni su **Elementary File** e **Dedicated File** ed è possibile instaurare comunicazioni cifrate tra microchip ed il computer (**Secure Messaging**).

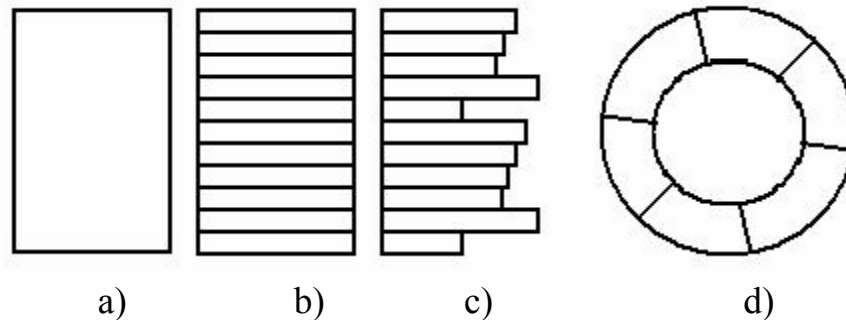
Un **Security Environment Object**, identificati da un **Security Environment Object** ID, è un contenitore logico di "meccanismi di sicurezza" (ossia algoritmi crittografici) che possono essere usati nei comandi crittografici (descritti più avanti nella sezione dedicata ai comandi avanzati). Più specificamente un **Security Environment Object** contiene un insieme di riferimenti a strutture dati che descrivono i meccanismi di sicurezza tra li quali, l'algoritmo crittografico, le modalità di esecuzione, le chiavi crittografiche e altre eventuali informazioni richieste dall'algoritmo specifico

3.5.2 Tipologia e formato di un file

Gli Elementary File sono semplici contenitori di dati. Possono essere in quattro tipi:

- **Transparent**: un file binario che contiene una semplice sequenza di byte. Sono usati tipicamente per memorizzare dati in formato binario. L'accesso al contenuto avviene in maniera analoga ad un file ad accesso casuale attraverso un puntatore relativo all'inizio del file codificato mediante un offset;
- **Linear Fixed**: un file composto da record a lunghezza fissa. Sono composti da record binari (al massimo 254) di lunghezza fissa. Ogni record può contenere al più 254 byte ed è identificato da un indice compreso tra 1 e 254 (dove 1 indica il primo record creato nel file). L'aggiornamento del file può avvenire in due modalità: **update** (aggiornamento di un record), **append** (aggiunta di un nuovo record in coda);
- **Linear Variable**: un file composto da record a lunghezza variabile. Sono composti da record binari (al massimo 254) di lunghezza variabile. Analogamente ai **LinearFixed**, ogni record può contenere al più 254 byte ed è identificato da un indice compreso tra 1 e 254. Tuttavia ogni record è preceduto da un byte che specifica la lunghezza dei record stesso. L'aggiornamento del file avviene in maniera analoga ai **Linear Fixed**.
- **Cyclic Fixed**: un file ciclico composto da un numero prestabilito di record a lunghezza fissa.

Sono composti da tiri numero prestabilito di record a lunghezza fissa. Analogamente ai **LinearFixed**, ogni record può contenere al più 254 byte ed è identificato da un indice compreso tra 1 e 254. Tuttavia la posizione logica del record all'interno del file viene assegnata in maniera diversa: il record con indice 1 corrisponde al record successivo all'ultimo acceduto, inoltre, quando il puntatore raggiunge l'ultimo record del file automaticamente viene spostato sui primo.



3.5.3 Permessi di accesso ai file

Ad ogni **Elementary File o Dedicated File** è possibile assegnare un insieme di permessi di accesso, detti tipicamente Access **Condition (AC)**, che specificano le condizioni pregiudiziali all'esecuzione di specifiche azioni, quali lettura, scrittura, cancellazione, ecc. Più specificamente un'Access **Condition** è legata ad un particolare **Security Data Object** e può essere vista come un flag, detto Access Right, che viene abilitato se il comando associato a tale **Security Data Object** (ad esempio un PIN associato al comando di verifica) viene eseguito con successo. Ciascuna smart card definisce per ciascuna tipologia di oggetto il proprio insieme di Access **Condition** (a titolo di esempio si faccia riferimento alle **Tabella A1, A2 e A3** nella appendice A che definiscono le AC associate a **Elementary File, Dedicated File** e **Security Data Object** della smart card virtuale fornita con l'emulatore).

A ciascun oggetto creato nella EEPROM della smart card vengono assegnati i valori per le Access Condition assegnate, ossia, viene associato a ciascuna AC l'ID di uno **Security Data Object** la cui verifica positiva abilita all'esecuzione dell'operazione relativa all'AC. In alternativa è possibile assegnare il valore 0x00 se l'operazione non deve essere mai consentita, o, viceversa, 0xFF per consentire in ogni caso l'operazione.

Il meccanismo di assegnazione delle Access Condition consente di definire diversi livelli di protezione dei **Dedicated File e Elementary File**. Ad esempio le operazioni di lettura possono essere subordinate alla verifica di un **Security Data Object** che rappresenta il PIN utente mentre le operazioni di scrittura, cancellazione e creazione di **Elementary File** può essere legata alla verifica di una chiave crittografica di amministrazione.

3.5.4 Secure Messaging

Il Secure Messaging (SM) consente di proteggere la comunicazione tra microchip applicazione, salvaguardandone la riservatezza, mediante algoritmi crittografici ed è utile soprattutto quando la trasmissione viene eseguita da un computer remoto (diverso da quello a cui è connesso il lettore di smart card). Gli algoritmi crittografici e le chiavi di cifratura sono definiti mediante *Security Data Object* e dipendono fortemente dalla potenza di calcolo dei microchip.

Solitamente si adottano algoritmi di cifratura DES, 3DES, RSA, ecc. Lo standard specifica che tutti i comandi possono essere inviati al microchip in chiaro o in modalità SM.

3.6 L'INSIEME DEI COMANDI

I comandi supportati dal microchip, implementati dal sistema operativo, sono anch'essi definiti nella parte 4 dello standard. L'insieme di tali comandi costituisce un *framework*, o, in altre parole, una sorta di API, comprendente funzioni di gestione del *file system* come creazione di file, lettura e scrittura di dati ecc., funzioni di verifica del PIN, di generazione di chiavi crittografiche, di firma digitale, ecc.

L'unità fondamentale nella comunicazione con il microchip è detta APDU (*Application Protocol Data Unit*). I comandi (sia quelli specificati nello standard ISO 7816, sia quelli proprietari dello specifico sistema operativo) e le risposte del sistema operativo sono codificati in APDU.

3.6.1 Command APDU

Un comando diretto al microchip è codificato in una *Command APDU* (Figura 3.4) paragonabile ad un record composto da sette campi, ciascuno di lunghezza 8 bit: CLA (*Class*), INS (*Instruction*), P1 (*Parameter 1*), P2 (*Parameter 2*), LC (*Length of the Command*), DATA (*data*), LE (*Length of Expected*). I primi quattro rappresentano *l'header* obbligatorio che specifica il particolare comando, i restanti identificano il corpo opzionale che contiene gli eventuali parametri di ingressi specifici del comando. Il byte CLA indica la classe di appartenenza del comando (ossia in quale sezione dello standard ISO 7816 è definito o se appartiene ad un insieme proprietario di comandi) e la modalità di invio (normale o SM). Il campo INS individua il comando all'interno nella classe specificata in CLA. P1 e P2 sono parametri aggiuntivi specifici del particolare comando. LC riporta la lunghezza del campo DATA che segue (può essere omesso se il campo DATA non è presente). DATA contiene i dati da elaborare mediante il comando (ad esempio i dati da scrivere in un file). Infine, LE indica la lunghezza attesa della risposta al comando inviata dal microchip.

Ogni Command APDU viene eseguita dal sistema operativo ed agisce sull'EF o il *Dedicated File* correntemente selezionato (allo start-up è automaticamente selezionato *il Master File*).

3.6.2 Response APDU

La risposta dei microchip ad un comando è codificata in una **Response APDU** composta da un corpo opzionale DATA seguito da due byte SW1 (**Status Word 1**) e SW2 (**Status Word 2**).

SW1	SW2	Descrizione
Esecuzione normale		
0x90	0x00	Comando eseguito con successo
0x61	0xXX	Comando eseguito con successo, SW2 indica il numero di byte della Response APDU ancora disponibili
Warning		
0x62	0x00	Warning generico
0x62	0x81	Dati non validi o corrotti
0x62	0x82	EOF raggiunto
0x62	0x83	Il file selezionato è stato invalidato
0x62	0x84	File Control Information (FCI) non valide
0 3	0x0	Errore generico
0x63	0x81	File pieno
0x63	0xCX	Il significato dell'errore dipende dal comando
Errori	di esecuzione	
0x64	0x00	Errore di esecuzione dei comando
0x65	0x00	Errore generico
0x65	0x81	Errore di memorizzazione
0x66	0xXX	Riservati per future estensioni
Errori	di verifica	
0x67	0x00	Lunghezza dei comando (LC) non valida
0x68	0x00	Comando CLA) non supportato
0x68	0x81	Canale non supportato
0x68	0x83	Modalità Secure Messaging non supportata
0x69	0x00	Comando non consentito
0x69	0x81	Il comando non è compatibile con la struttura dei file
0x69	0x82	Accesso non consentito (permessi di accesso non concessi)
0x69	0x83	Security Object bloccato
0x69	0x84	I dati dei comando non sono validi
0x69	0x85	Condizioni di uso non soddisfatte
0x69	0x86	Comando non valido perché non c'è alcun file selezionato
0x69	0x87	Oggetto Secure Messaging non trovato
0x68	0x88	Oggetto Secure Messaging non valido
0x6A	0x00	Campo P1 o P2 non corretto
0x6A	0x80	Parametri nel campo DATA non corretti
0x6A	0x81	Funzione non supportata
0x6A	0x82	File non trovato
0x6A	0x83	Record non trovato

0x6A	0x84	Non c'è abbastanza memoria disponibile nel file o nella memoria
0x6A	0x85	Campo LC inconsistente con la struttura TLV
0x6A	0x86	Campo P1 o P2 non corretto
0x6A	0x87	Campo LC inconsistente con i campi P1 e P2
0x6A	0x88	Oggetto non trovato
0x6B	0x00	Campi P1 e P2 non corretti
0x6C	0xXX	Campo LE non corretto. XX indica le dimensioni corrette dei campo DATA della Response APDU
0x6D	0x00	Campo INS non valido o non supportato
0x6E	0x00	Campo CLA non valido o non supportato
0x6F	0x00	Errore interno

SW1 e SW2 riportano lo stato finale dell'elaborazione, ossia specificano se l'elaborazione è stata conclusa con successo o se si è verificato un errore. In particolare i valori SW1 = 0x90 e SW2 = 0x00 indicano che il comando è stato eseguito con successo. In questo caso il campo DATA contiene il risultato dell'elaborazione (se il comando ne prevede uno).

Se la risposta al comando contiene più byte di quelli aspettati (ossia è più lunga del valore specificato nel campo LE) SW1 viene impostato al valore 0x91 mentre SW2 viene impostato al numero di byte ancora disponibili. Valori diversi di SW1 indicano una condizione di errore (l'insieme dei codici di errore definiti nella parte 4 dello standard è riportato nella **Tabella 3.2**).

3.7 CICLO DI VITA

Il ciclo di vita della smart card virtuale prevede due stati: *unformatted* e *formatted*.

Nello stato *unformatted* la memoria EEPROM della smart card è vuota e la maggior parte dei comandi sono disabilitati. Per portare la carta nello stato *formatted* esiste uno speciale comando di formattazione che inizializza la EEPROM, crea il *Master File* e porta la smart card nello *stato formatted* predisponendola all'uso.

Implementazioni

Le smart card sono state integrate già in numerose applicazioni e su diverse piattaforme, alcune delle quali sono usate quotidianamente da molte persone (ad esempio nei cellulari GSM).

Questa parte della tesina tratta le principali integrazioni partendo dalla piattaforma Windows dove, grazie alle smart card, è possibile eseguire procedure di Strong Authentication, firmare e cifrare la posta elettronica, ecc. per arrivare al GSM e alle specifiche EMV disegnate per realizzare applicazioni bancarie.

4.1 LE SMART CARD SULLE PIATTAFORME MICROSOFT

Le piattaforme Microsoft, da Windows 95 in poi, implementano il supporto per le smart card tramite le specifiche PC/SC. La comunicazione con il lettore è attuata mediante un driver che esporta l'interfaccia dell'IFD *Handler* mentre l'invio dei comandi alla smart card può avvenire sia mediante l'API del *Resource manager*, sia tramite il *Service Provider*. Le funzioni crittografiche della smart card sono invece accessibili attraverso un modulo separato detto *CryptoAPI* che offre tutti i principali algoritmi crittografici indipendentemente da quale sia il dispositivo hardware o la libreria software che dà l'effettiva implementazione.

In questo capitolo si darà una breve descrizione di tale architettura e si mostrerà come Windows 2000, Windows XP Outlook e Internet Explorer possono essere potenziati sfruttando le funzionalità crittografiche della smart card.

4.1.1 CryptoAPI

CryptoAPI è un modulo del sistema operativo composto da alcune dll che fornisce i servizi crittografici utilizzati dalle applicazioni per cifrare/decifrare insiemi di dati, per apporre/verificare firme elettroniche, ecc. L'interfaccia di programmazione che propone consente di realizzare applicazioni crittografiche senza la necessità di conoscere la reale implementazione degli algoritmi crittografici che può essere totalmente realizzata in una libreria software separata o può poggiare sulle funzionalità offerte da una smart card.

La figura nella pagina successiva mostra l'architettura generale. Come evidenziato nel disegno, le funzioni fornite da *CryptoAPI* possono essere classificate in tre categorie: funzioni di gestione dei certificati, funzione di gestione dei messaggi crittografici e funzioni crittografiche.

Le prime consentono di maneggiare i certificati digitali associati alle chiavi crittografiche attraverso funzioni di creazione, verifica, codifica, decodifica ed estrazione dei dati.

Le seconde permettono di gestire i messaggi crittografati (ossia cifrati e/o firmati) in formato PKCS#7 tramite funzioni di creazione, cifratura e firma di messaggi.

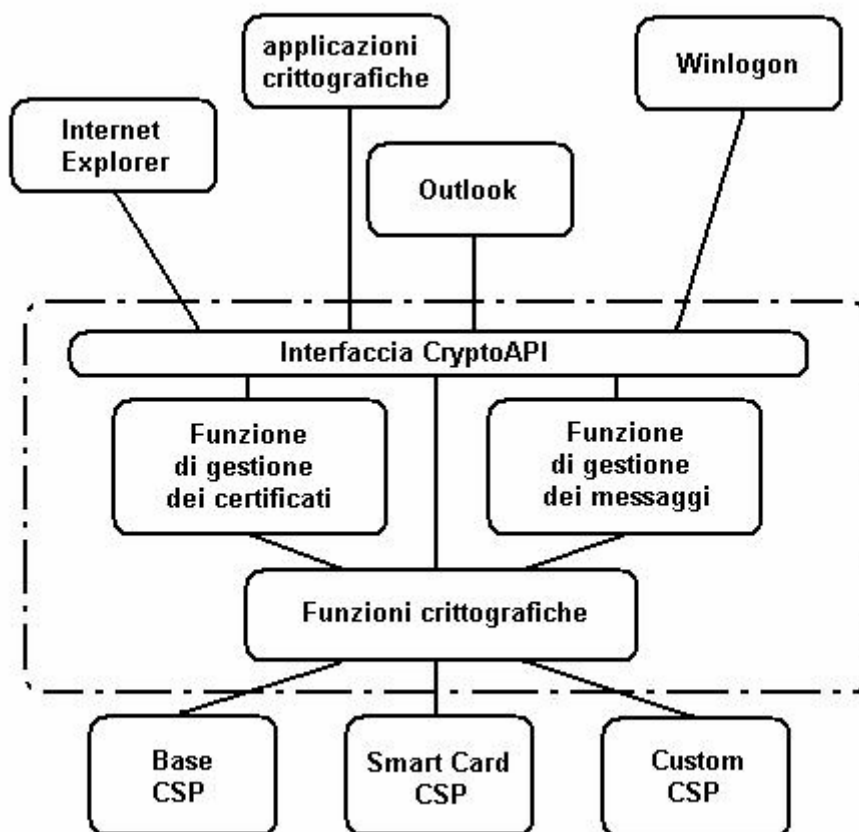
Le ultime, infine, rappresentano le funzioni crittografiche vere e proprie sulle quali poggia l'implementazione di quelle appartenenti alle altre categorie.

Le funzioni crittografiche esportate da *CryptoAPI* sono in realtà implementate in librerie esterne indipendenti chiamate *Cryptographic Service Provider (CSP)* che possono eventualmente coinvolgere una smart card, o un generico dispositivo crittografico esterno,

per espletare le proprie funzioni. Il grafico riportato nella pagina seguente, mostra i diversi CSP forniti dalle varie versioni di Windows mostrando le loro principali caratteristiche.

E' a questo livello che si realizza l'integrazione di una smart card crittografica su piattaforme Windows. Più specificamente, per usare le funzionalità crittografiche di una particolare smart card all'interno dell'architettura CryptoAPI, e quindi integrare "nativamente" tale smart card nell'infrastruttura di sicurezza di Windows, occorre realizzare uno specifico CSP e installarlo correttamente nel sistema operativo

Architettura generale di CryptoAPI



Le applicazioni che usano le funzioni dell'interfaccia CryptoAPI, quali ad esempio Internet Explorer e Outlook, non si preoccupano di come e dove gli algoritmi crittografici siano implementati ma si limitano ad impostare il CSP che soddisfa le loro necessità e a chiamare le funzioni dell'interfaccia CryptoAPI.

4.1.1.1 Internet Explorer

Sii Internet Explorer le funzionalità crittografiche della smart card sono usate, tramite le CryptoAPI, per realizzare la Strong Authentication dei client mediante firma digitale nelle

connessioni con protocollo HTTPS implementato a partire dal protocollo SSLv3 (Secure Socket Layer versione 3).

Più specificamente, quando un sito web protetto tramite protocollo SSL, richiede l'autenticazione dei client, Internet Explorer richiama le funzioni di firma digitale fornite da CryptoAPI che, a sua volta, chiama le corrispondenti funzioni del CSP selezionato per l'autenticazione.

4.1.1.2 Outlook

Su Outlook 98, 2000, XP e Outlook Express 4, 5 e 6 le funzionalità crittografiche della smart card sono usate, sempre tramite le CryptoAPI, per proteggere la posta elettronica tramite cifratura e firma digitale delle email.

4.1.1.3 Smart card logon su Windows 2000/XP

Per garantire maggiore sicurezza nell'accesso alla postazione, Windows 2000 e Windows XP consentono di sostituire il classico meccanismo di logon basato sulla coppia **user-password** con un protocollo di autenticazione più sicuro basato su firma digitale mediante smart card.

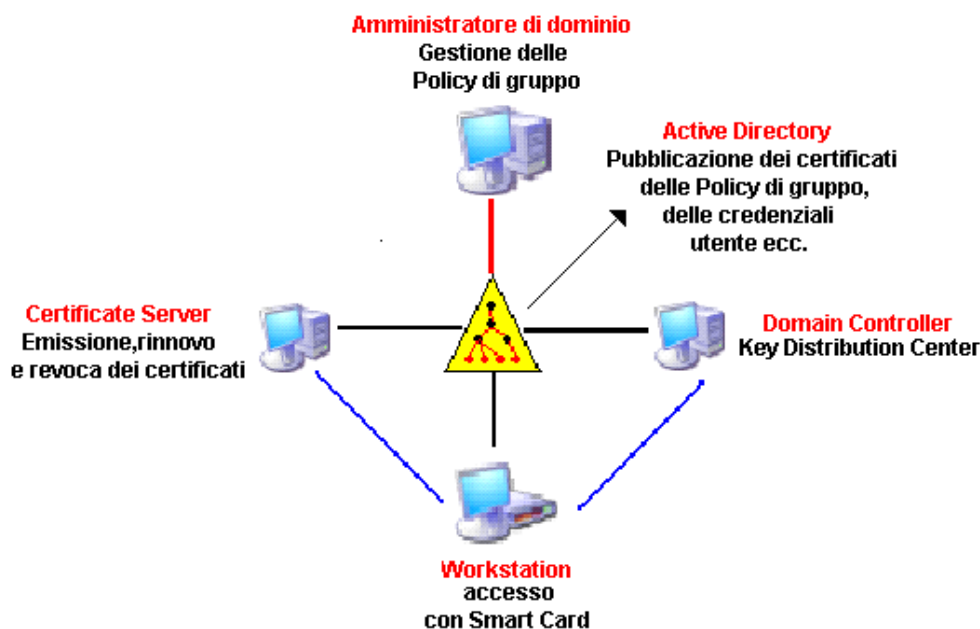
Poiché la chiave privata con cui viene apposta la firma digitale è conservata su smart card ed è protetta da PIN tale protocollo è invulnerabile (la attacchi mediante "sniffer" o dei tipo "Cavallo di Troia" mirati all'appropriazione delle credenziali utente (che in questo caso sono rappresentate dalla coppia certificato digitale-chiave privata).

4.2 LA PROCEDURA DI ACCESSO AL SISTEMA

L'usuale procedura di **logon** (ovvero quella adottata dai principali Sistemi Operativi quali Windows 2000/NT, Unix/Linux, Novell Netware, ecc.) è fondata sulla coppia **user-password**. Ci sono però due aspetti che la rendono vulnerabile. In primo luogo, anche ipotizzando una **password** estremamente complicata, grazie alla potenza di calcolo degli attuali PC il tempo che occorre a completare l'attacco di forza bruta (cioè per tentativi fino alla scoperta della **password** corretta) non è eccessivamente elevato; se poi, come solitamente avviene, la **password** è scelta con un valore mnemonico, come ad esempio una data di nascita, o un soprannome ecc. lo spazio delle possibili soluzioni, e quindi il tempo di calcolo, si riduce notevolmente. In secondo luogo, tale meccanismo è sensibile ad attacchi mirati alla appropriazione delle coppie **user-password**, compiuti ad esempio con uno "sniffer" sul canale di comunicazione tra **workstation** e **domain controller** o da un "Cavallo di Troia" il quale, una volta entrato nella **workstation** è in grado di intercettare le password digitate.

Molti sistemi operativi offrono la possibilità di "personalizzare" la procedura di **logon** al fine di realizzare un meccanismo di autenticazione più sicuro. Windows NT/2000 ad esempio consente di sostituire il tradizionale modulo di **logon** (chiamato GINA, **Graphical Identification and Authentication**) con un modulo realizzato seguendo specifiche definite

da Microsoft. Unix/Linux, in maniera del tutto simile a Windows, adotta un sistema chiamato PAM (**Pluggable Authentication Module**) cui possono essere aggiunti moduli di autenticazione personalizzati. Novell fornisce un sistema più complesso ma più potente chiamato NMA (**Novell Modular Authentication System**) che consente di realizzare diversi metodi di autenticazione che possono essere eventualmente usati in sequenza (ovvero l'utente è autenticato solo dopo che è stato autenticato da tutti i metodi in sequenza).



In tali sistemi operativi la personalizzazione della procedura di logon, implica comunque la realizzazione di uno o più moduli software seguendo le specifiche definite dalla azienda produttrice. Pertanto, ciò richiede un notevole sforzo indirizzato allo sviluppo dei software. In Windows 2000 invece la procedura di autenticazione basata su un algoritmo di tipo challenge-response con firma digitale mediante smart card è integrata nel sistema e non richiede ulteriori sviluppi. il paradigma generale è il seguente: il sistema genera una stringa casuale, detta **challenge**, e la presenta all'utente; quest'ultimo mediante la chiave privata conservata sulla smart card appone al challenge la propria firma digitale (dopo aver inserito il PIN che abilita la smart card ad eseguire l'operazione di firma) producendo una nuova stringa detta **response**; il sistema riceve tale response e verifica la firma apposta mediante il certificato digitale associato all'utente. Se la verifica ha successo l'utente è autenticato. Il protocollo challenge-response è considerato estremamente sicuro poiché il challenge è generato casualmente ad ogni accesso, pertanto, se anche uno "sniffer" lo intercettasse durante la comunicazione, sarebbe influenzabile per un successivo tentativo di autenticazione. Inoltre, a differenza dei protocolli basati su user-password, l'informazione segreta associata all'utente, ovvero la chiave privata, è conservata solo, ed esclusivamente sulla smart card e nessun "Cavallo di Troia" potrà mai accedervi.

4.3 ACCESSO AI SISTEMI WINDOWS 2000 CN SMART CARD

Windows 2000 consente l'autenticazione mediante smart: card in tre differenti modalità: accesso interattivo al dominio, ovvero autenticazione dell'utente da una workstation collegata

ad tini LAN; accesso remoto, inteso come autenticazione remota dell'utente tramite un servizio RAS; accesso a risorse web ovvero autenticazione all'interno di una navigazione nel web durante l'accesso ad un sito protetto con il protocollo SSI

In tutti i casi è richiesta l'installazione di una Public Key Infrastructure per la gestione delle chiavi e dei certificati associati agli utenti. Più specificamente per realizzare in Windows 2000 una PKI che consenta il logon con smart card occorre installare **Microsoft** Certificate Server che funge di **Certification Authority (CA)** ed ha l'onere di emettere, rinnovare e revocare i certificati degli utenti, Active Directory, che agisce come **Directory Service (DS)**, dove sono pubblicati i certificati, le liste di revoca, le policy di gruppo, i profili utente ecc. e un domain controller. L'amministratore di dominio opera direttamente su Active Directory per la gestione dei profili utenti e dei relativi certificati e delle policy utente e di gruppo. In particolare l'amministratore può impostare delle regole (policy) per il controllo dell'uso della smart card al fine di impostare quali utenti o gruppi di utenti devono necessariamente accedere al dominio con la smart card e quali invece possono entrare anche (o solo) con la coppia user-password

Il meccanismo di autenticazione con smart card di Windows 2000 si basa sull'estensione PKINIT del protocollo standard Kerberos V5. Tale protocollo fornisce un meccanismo di tipo **challenge-response** per attuare la mutua autenticazione tra client e server e richiede una componente speciale detta **Key Distribution Center (KDC)** associata al **domain controller**. Il KDC, cui sono demandate tutte le richieste di autenticazione al sistema, ha il compito di verificare la firma apposta al **challenge** (e di conseguenza la bontà del certificato inviato dall'utente) e di estrarre da **Active Directory** le informazioni riguardanti l'utente (gruppo di appartenenza, permessi, ecc.) per consentirgli l'accesso alle sole risorse cui è abilitato così come definito nella policy.

4.3.1 Configurare il dominio Windows 2000

Per abilitare l'accesso al dominio mediante smart card occorre installare e configurare le componenti mostrate in nella figura precedente. Inoltre, le **workstation** devono essere corredate di lettore smart card aderente alle specifiche PC/SC e gli utenti devono essere in possesso di una delle smart card supportate dal **logon** di Windows 2000 (fornite da Gemplus, Siemens e Schlumberger).

4.3.2 Configurare Certificate Server

Certificate Server rilascia certificati per usi differenti (come ad esempio firma digitale, protezione e-mail ecc.) basati su **template** pubblicati da **Active Directory**. In particolare per emettere certificati per il logon con smart card occorre configurare **Certificate Server** in

modo che possa rilasciare certificati per *Enrollment Agent* e *Smart Card Logon*. A tale fine, sulla macchina Windows 2000 su cui è installato *Certificate Server* occorre operare come segue

- Aprire la console di *Certificate Server* dal menù **Start/Program/Administration Tools**
- Aprire la directory **Policy Settings**. Tale directory contiene tutti i tipi di certificato che la CA può emettere.
- Premere il tasto destro e selezionare New e di seguito Certificate to issue Selezionare Enrollment Agent e Smart Card Logon

4.3.3 Specificare la policy

Per rilasciare certificati agli utenti del dominio è necessario impostare la policy. In altre parole occorre specificare quali utenti sono abilitati a richiedere certificati. Si opera nel modo illustrato di seguito:

- Aprire la console Active Directory Sites and Services dal menù Start/Program/Administration Tools
- Aprire la directory Public Key Services e selezionare Certificate Templates
- Selezionare la voce Enrollment Agent, premere il tasto destro e scegliere Properties
- Nella sezione Security abilitare i permessi di Read ed Enroll per i gruppi o gli utenti che devono agire come Enrollment Agent
- Ripetere i punti 4 e 5 selezionando i template MachineEnrollmentAgent per abilitare i permessi di Read ed Enroll dei gruppi o degli utenti che devono agire come Machine Enrollment Agent
- Ripetere i punti 4 e 5 selezionando i template SmartCardLogon per abilitare i permessi di Read ed Enroll dei gruppi degli utenti che devono usare la smart card.

4.3.4 Configurare l'Enrollment Station

Per Enrollment Station si intende una macchina corredata di lettore smart card mediante la quale è possibile richiedere alla CA un certificato per smart card. Per configurare l'Enrollment Station occorre definire un Enrollment Agent (un utente speciale che può generare certificati su smart card a beneficio di altri

utenti). A tale scopo è necessario emettere un certificato per l'utente prescelto, basato sul *template EnrollmentAgent* sopra menzionato, nel seguente modo:

- Aprire *Microsoft Management Console* ed aggiungere lo *snap-in* Certificates
- Aprire la directory Personal, premere il tasto destro nella parte destra della finestra e selezionare All Tasks/Request New Certificate
- Riempire i campi e, quando richiesto, selezionare il *template* Enrollment Agent in questo modo *l'Enrollment Station* e *l'Enrollment Agent* sono pronti a rilasciare i certificati agli utenti che necessitano di smart card.

4.3.5 Emissione dei certificati su smart card

Per emettere un certificato sulla smart card operare come segue:

Dalla *Enrollment Station* connettersi all'indirizzo: `http://<CA Server>/certsrv` (dove CA Server è il nome del server su cui è installata la CA)
Sulla pagina di benvenuto selezionare Request Certificate e premere Next
Selezionare Advanced Request e premere Next

Nella pagina successiva selezionare Request a certificate for a Smart Card e premere Next

Riempire la form di richiesta con le informazioni opportune. In particolare occorre impostare il template del certificato (con il valore Smartcard *Logon*), il nome della CA, il Cryptographic Service Provider relativo alla smart card in uso, il certificato *dell'Enrollment Agent* e l'utente per cui si richiede il certificato.

Inserire una smart card vuota nel lettore e premere Enroll

A questo punto la smart card è pronta per il *logon* da qualsiasi workstation appartenente al dominio.

Crittografia

5.1 LA CRITTOGRAFIA

La crittografia si propone di *ricercare algoritmi* capaci di proteggere con un considerevole grado di sicurezza le informazioni ad alto valore contro possibili attacchi da parte di

criminali della concorrenza o di chiunque possa usarle per arrecare danno. Comprende tutti gli aspetti relativi alla sicurezza dai messaggi all'autenticazione degli interlocutori, alla verifica dell'integrità.

Sistemi crittografici sono presenti un pò dovunque nella storia anche se, finora, sono stati usati principalmente in ambito militare. Già Giulio Cesare, quando inviava un messaggio riservato, non fidandosi del messaggero sostituiva ad ogni A una I, ad ogni B una E e così via per le altre lettere. Solo chi conosceva la regola "Sposta di tre" poteva decifrare il messaggio. Inoltre, durante la seconda guerra mondiale alcuni successi alleati sono riconducibili alla scoperta della regola di cifratura usata dai tedeschi per nascondere i loro messaggi

Attualmente la crittografia non è più circoscritta all'ambito strettamente militare. Si sta cercando di sfruttarne tutti i vantaggi per creare una rete sicura a disposizione della società moderna che richiede, ogni giorno, algoritmi sempre più potenti e sicuri per la protezione delle informazioni e per l'autenticazione degli utenti.

Terminologia Per meglio familiarizzare con i concetti nell'ambito della crittografia diamo alcune definizioni:

- **la Crittografia** è definita come l'arte o la scienza di rendere segreti i messaggi;
- **l'analisi crittografica** è Parte di violare un sistema crittografico e decifrarne i messaggi
- **la crittologia** è il ramo della matematica che studia i fondamenti matematici della crittografia.
- **il messaggio in chiaro** è chiamato plaintext o cleartext;
- **il messaggio cifrato** è detto ciphertext. Esso si presenta come una sequenza casuale di simboli ed è quindi incomprensibile;
- **la cifratura e decifrazione** sono rispettivamente le trasformazioni del messaggio da plaintext a ciphertext a viceversa. Esse avvengono solitamente per mezzo di una chiave. In tal caso la decifrazione può avvenire solo se si conosce la chiave usata nella cifratura;
- **l'autenticazione è la verifica dell'identità** di un individuo coinvolto in una comunicazione;
- **la verifica dell'integrità** è la prova che il messaggio non ha subito modifiche durante la trasmissione;
- **la firma digitale** è una stringa cifrata ricavata dal messaggio che consente di identificare il mittente e di verificare l'integrità del messaggio;
- **un certificato digitale** è un documento di identità virtuale che consente di identificare entità sulla rete.

La crittografia fornisce una serie di algoritmi e di metodi per rendere il messaggio indecifrabile. Alcuni di essi sono molto potenti ed hanno resistito ai più svariati attacchi, altri meno sicuri ma altrettanto importanti.

L'obiettivo di ogni algoritmo di cifratura è quello di rendere il più complicato possibile la decifratura di un messaggio senza la conoscenza della chiave. Se l'algoritmo di cifratura è buono, l'unica possibilità per decifrare il messaggio è provare, una per volta, tutte le possibili chiavi fino a trovare quella giusta, ma tale numero cresce esponenzialmente con la lunghezza della chiave. Quindi, se la chiave è lunga soltanto 40 bit saranno necessari al più 211 differenti tentativi.

Se ne deduce che l'operazione più delicata in un sistema crittografico è proprio la generazione della chiave.

Affinché esso sia effettivamente sicuro, deve prevedere chiavi di considerevole lunghezza; inoltre, le chiavi devono essere generate in maniera realmente casuale e dunque assolutamente imprevedibile per un ipotetico decrittatore. Per tale motivo vengono scartati i generatori di numeri pseudo-casuali forniti dal computer, di solito adottati per giochi e simulazioni, e si preferisce adottare sistemi più complessi che sfruttano il rumore di fondo del mondo fisico che non può in nessun modo essere predetto. Buone sorgenti di numeri casuali risultano essere i processi di decadimento radioattivo, il rumore di fondo in un semiconduttore, o gli intervalli di tempo che intercorrono tra le azioni dell'operatore sulla tastiera. La sorgente più comunemente utilizzata sfrutta il movimento del mouse o la misura in millisecondi del tempo di battitura dell'operatore.

Risulta tuttavia molto difficile determinare la bontà di un algoritmo. Talvolta algoritmi che sembravano molto promettenti si sono rivelati estremamente semplici da violare lanciando l'opportuno attacco. E' comunque preferibile affidarsi a quegli algoritmi che sembrano resistere da più tempo

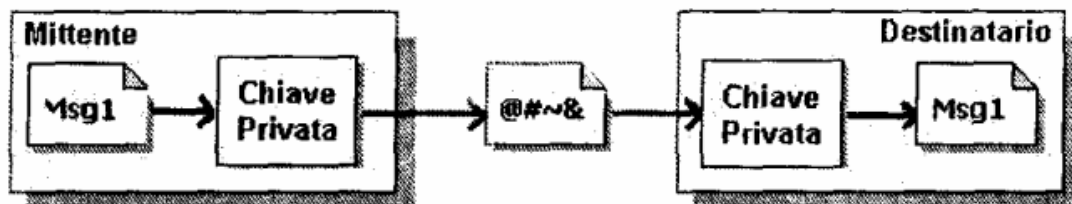
Gli algoritmi di cifratura sono suddivisi in due classi:

- algoritmi a chiave privata, o algoritmi simmetrici;
- algoritmi a chiave pubblica.

La differenza è che i primi usano la stessa chiave per la cifratura e la decifratura, mentre i secondi usano due chiavi differenti, una pubblica e una privata, ma complementari.

5.2 ALGORITMI A CHIAVE PRIVATA

Gli algoritmi a chiave privata, o algoritmi simmetrici sono i più comunemente utilizzati. Essi usano la stessa chiave per cifratura e decifratura. Entrambi gli interlocutori conoscono la chiave usata per la cifratura, detta chiave privata, o chiave simmetrica, e soltanto loro possono cifrare e decifrare il messaggio (come in figura).



Sulla base del tipo di computazione si individuano due tipi di cifratura:

- **stream cipher (cifratura sequenziale)**: Il messaggio è visto come una sequenza di bit e viene cifrato un bit alla volta. Sono sicuramente i più veloci ma sono considerati poco sicuri, sebbene la sicurezza dipenda dall'algoritmo utilizzato;
- **block cipher (cifratura a blocchi)**: Il messaggio è suddiviso in blocchi di lunghezza fissa e cifrato un blocco per volta. Sebbene siano più lenti dei precedenti, sono considerati più sicuri perché ogni blocco è cifrato mescolandolo opportunamente al blocco precedente;

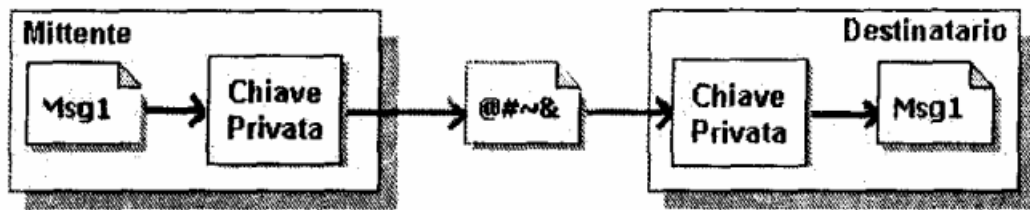
Tra i primi menzioniamo:

- **RC4**, sviluppato da RSA Data Security Inc., è un algoritmo molto veloce che accetta chiavi a lunghezza variabile. La sua sicurezza non è ancora ben accertata ma, finora, ha resistito molto bene a diversi tipi di attacchi. Usa essenzialmente un generatore di numeri casuali, il numero generato è poi applicato mediante XOR alla sequenza di bit.

Tra i secondi menzioniamo:

- **DES**, sviluppato nel 1970, è divenuto poi lo standard per il governo americano. Adotta blocchi da 64 bit ed una chiave a 56 bit. Data la ridotta lunghezza della chiave risulta essere facilmente violabile dai computer moderni. Recentemente è stata sviluppata una variante detta **Triple-DES** o **3DES** che cifra il messaggio tre volte con altrettante differenti chiavi.

- **Blowfish**, sviluppato da Bruce Schneier. Adotta un blocco a 64 bit ed una chiave fino a 448 bit. Ha ottenuto grandi consensi da parte della comunità internazionale e finora non si conoscono attacchi vincenti. Usato in alcuni famosi sistemi come **Nautilus** e **PGPfone**



- **IDEA (International Data Encryption Algorithm)**, sviluppato in Svizzera, usa una chiave a 128 bit ed è generalmente considerato molto sicuro.
- **Rijndael**, altrimenti detto **AES (Advanced Encryption Standard)**, sviluppato recentemente da Joan Daemen and Vincent Rijmen (da cui il nome), usa chiavi di 128, 192 o 256 bit ed è attualmente considerato il più potente tra gli algoritmi **block cipher**

Gli algoritmi a chiave privata hanno il vantaggio di essere molto veloci, idonei per cifrare grandi volumi di dati, ma hanno lo svantaggio di richiedere la distribuzione della chiave privata a tutti i destinatari. Necessitano quindi di un ulteriore canale sicuro attraverso cui distribuire la chiave. Tale contraddizione, nel recente passato, ha posto dei limiti allo sviluppo della crittografia fino all'introduzione degli algoritmi a chiave pubblica.

5.3 ALGORITMI A CHIAVE PUBBLICA

Gli algoritmi a chiave pubblica usano (due chiavi complementari, dette **chiave pubblica** e **chiave privata**, create in modo che la chiave privata non può assolutamente essere ricavata dalla chiave pubblica.

il paradigma di comunicazione è il seguente:

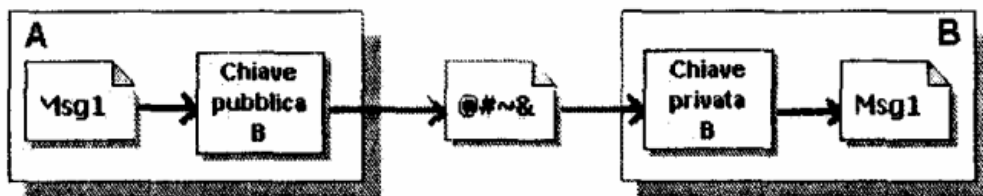
i due interlocutori A e B hanno entrambi una coppia di chiavi. A richiede a B la sua chiave pubblica con la quale cifra il messaggio e spedisce il risultante messaggio cifrato a B.

Il messaggio cifrato con una chiave pubblica può essere decifrato solo con la corrispondente chiave privata. Pertanto B, mediante la sua chiave privata, può decifrare il messaggio e leggerlo in tutta sicurezza.

Con questo metodo solo la chiave privata deve essere tenuta segreta mentre la chiave pubblica può essere distribuita a chiunque voglia spedire un messaggio al possessore della chiave. Qualora finisse nelle mani di un pirata, egli non potrà fare altro che cifrare messaggi senza poterli poi decifrare.

La crittografia a chiave pubblica si basa su algoritmi molto astuti facilmente calcolabili in un senso ma estremamente complicati da risolvere in senso inverso. L'algoritmo di

Diffie-Hellman, ad esempio, si basa sul principio dei logaritmi. I due interlocutori A e B hanno ciascuno un numero segreto, rispettivamente x e y ed entrambi conoscono un numero pubblico g . A calcola g^x ed invia il risultato a B che può calcolare il valore $(g^x)^y$. Tale numero è comune ad entrambi (allo stesso modo A può calcolare $(g^y)^x$) e diventa pertanto la loro comune chiave segreta di cifratura. Qualcuno potrebbe obiettare e dire che intercettando g^x e g^y , e conoscendo g è possibile risalire a $(g^x)^y$, allora, per scoraggiare anche la più astuta delle spie,



A usa la funzione modulo e invia a B il valore $g^x \text{ mod } p$, poichè, come la matematica suggerisce, è quasi impossibile ricavare x da $g^x \text{ mod } p$ anche conoscendo g e p . Successivamente, l'algoritmo di Diffie-Hellman è stato perfezionato con ulteriori integrazioni matematiche dando origine ai sistemi crittografici moderni basati su una coppia di chiavi complementari, una privata, il valore di x , e una pubblica composta da g , p , e dal valore di $g^x \text{ mod } p$.

Gli algoritmi più diffusi sono:

- **Diffie-Hellman**, di cui sopra, è generalmente considerato sicuro, soprattutto nello scambio di chiavi simmetriche quando usato con chiavi abbastanza lunghe preferibilmente di almeno 1024 bit.
- **RSA**, è il più usato, sia per cifratura di messaggi che per apporre la firma digitale.

È generalmente considerato sicuro quando usato con chiavi di almeno 1024 bit (512 insicuro, 768 moderatamente sicuro, 1024 sicuro). È basato sulla difficoltà di scomporre un numero nei suoi fattori primi. Difatti presi due numeri sufficientemente grandi x e y , è estremamente semplice calcolare il loro prodotto $z = x * y$ ma risulta estremamente complicato scomporre z ricavando proprio i due numeri di partenza.

- **Curve ellittiche** è un algoritmo emergente relativamente giovane, ma molto lento. È considerato estremamente sicuro ma non è stato ancora sottoposto agli stessi test a cui è stato sottoposto RSA.
- **DSS (Digital Signature Standard)** usato principalmente per la firma digitale è stato adottato dal governo USA.

5.4 LA TECNICA ADOTTATA NELLA PRATICA

Gli algoritmi a chiave pubblica sono considerevolmente più lenti di quelli a chiave privata, in particolare nella cifratura di grosse moli di dati. Pertanto, nei sistemi crittografici si preferisce adottare algoritmi simmetrici per la cifratura dei messaggi e algoritmi a chiave pubblica per la cifratura delle chiavi simmetriche. Il mittente genera una chiave simmetrica, cifra il messaggio, cifra la chiave generata con la chiave pubblica del destinatario e invia insieme il messaggio e la chiave generata. Il destinatario decifra la chiave simmetrica con la propria chiave privata ed infine decifra il messaggio.

5.5 LA FIRMA DIGITALE

Grazie alla complementarità delle chiavi pubblica e privata, una stringa cifrata con una chiave può essere decifrata solo mediante l'altra chiave. Pertanto, la decifratura di tiri testo mediante una chiave assicura che esso è stato cifrato con la chiave complementare

Gli algoritmi di firma digitale sfruttano questa caratteristica per verificare la reale provenienza del messaggio (autenticazione del mittente).

La firma digitale è una stringa ricavata dal messaggio applicando un particolare algoritmo, cifrata mediante la chiave privata del mittente e spedita insieme al messaggio. La decifratura della firma mediante la chiave pubblica prova che è stata cifrata dal mittente o da qualcuno in possesso della sua chiave privata. Inoltre, il confronto della stringa decifrata con una stringa ricavata ex-novo dal messaggio applicando lo stesso algoritmo, consente di verificare l'integrità: se le due stringhe coincidono il messaggio è integro.

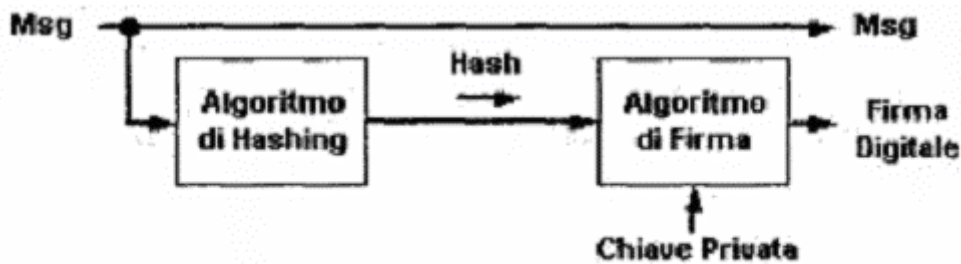
Il paradigma è il seguente: il mittente, unico possessore della chiave privata, produce un'impronta del messaggio, detta *hash, o message-digest*, e la cifra con la sua chiave privata. La stringa di *hash* cifrata rappresenta la firma digitale.

Il destinatario riceve il messaggio insieme alla firma: Dal messaggio, eventualmente cifrato, ricostruisce l'impronta, mentre dall'*hash*, dopo averlo decifrato con la chiave pubblica del mittente, ricava l'impronta del messaggio com'era al momento della sua spedizione. Se le due impronte così ottenute coincidono si è certi che la firma è stata apposta mediante la chiave privata del mittente e che il messaggio non è stato modificato durante la trasmissione.

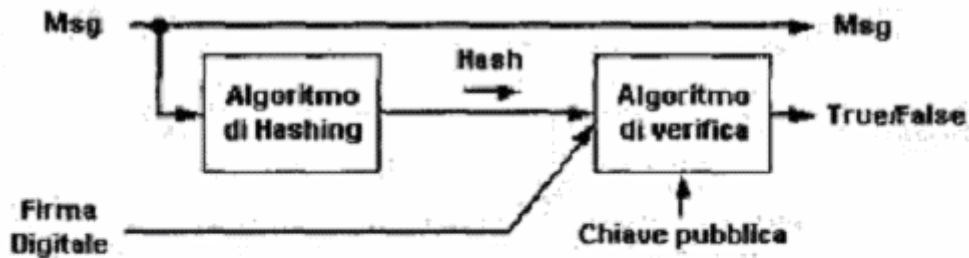
5.6 GLI ALGORITMI DI HASHING

Gli algoritmi di firma digitale si affidano principalmente sulla potenza degli algoritmi di hashing. Essi sono algoritmi *one-way* che producono, a partire da una stringa a lunghezza variabile, una stringa a lunghezza fissa (tipicamente tra 64 e 255 bit) che è caratteristica della stringa data.

La loro potenza è dovuta alle seguenti peculiarità: data una stringa di hash è computazionalmente impossibile ricavare il messaggio dal quale è stata generata; è computazionalmente impossibile determinare due messaggi che producono la stessa stringa di hash; qualsiasi messaggio, sottoposto allo stesso algoritmo qualsivoglia numero di volte produce sempre lo stesso valore di hash.



Decifratura della firma mediante chiave pubblica: lato mittente



Decifratura della firma mediante chiave pubblica: lato destinatario

Gli algoritmi più diffusi sono:

- **MD5 (Message Digest Algorithm 5)**, sviluppato da RSA Data Security inc., è il successore di MD2, e MD4, algoritmi ormai in disuso. Produce hash di 128 bit da stringhe di lunghezza arbitraria, è largamente usato ed è considerato ragionevolmente sicuro.
- **SHA (Secure Hash Algorithm)**, sviluppato dal NIST (*National Institute of Standards and Technology*) e dal NSA (*National Security Agency*), è usato dal governo USA e produce stringhe di hash a 160 bit da stringhe di lunghezza arbitraria. È considerato abbastanza sicuro. Usato solitamente insieme al DSS,

5.7 LA CERTIFICAZIONE

I certificati digitali svolgono una funzione essenziale nella crittografia a chiave pubblica. Il loro obiettivo è di autenticare un individuo certificando che la chiave pubblica, in esso contenuta, appartiene realmente al soggetto per il quale è stato rilasciato. Assume pertanto un ruolo determinante nello scambio della chiave pubblica.

Un certificato è, a tutti gli effetti, un documento di identità digitale. Come un documento reale, esso contiene un insieme di attributi che identificano il possessore del certificato ed è rilasciato da un organismo ufficiale, eletto **Autorità di Certificazione**, ufficialmente riconosciuto dalla società, che garantisce l'autenticità delle informazioni in esso contenute. Solitamente, oltre alle informazioni relative al soggetto, contiene la sua chiave pubblica,

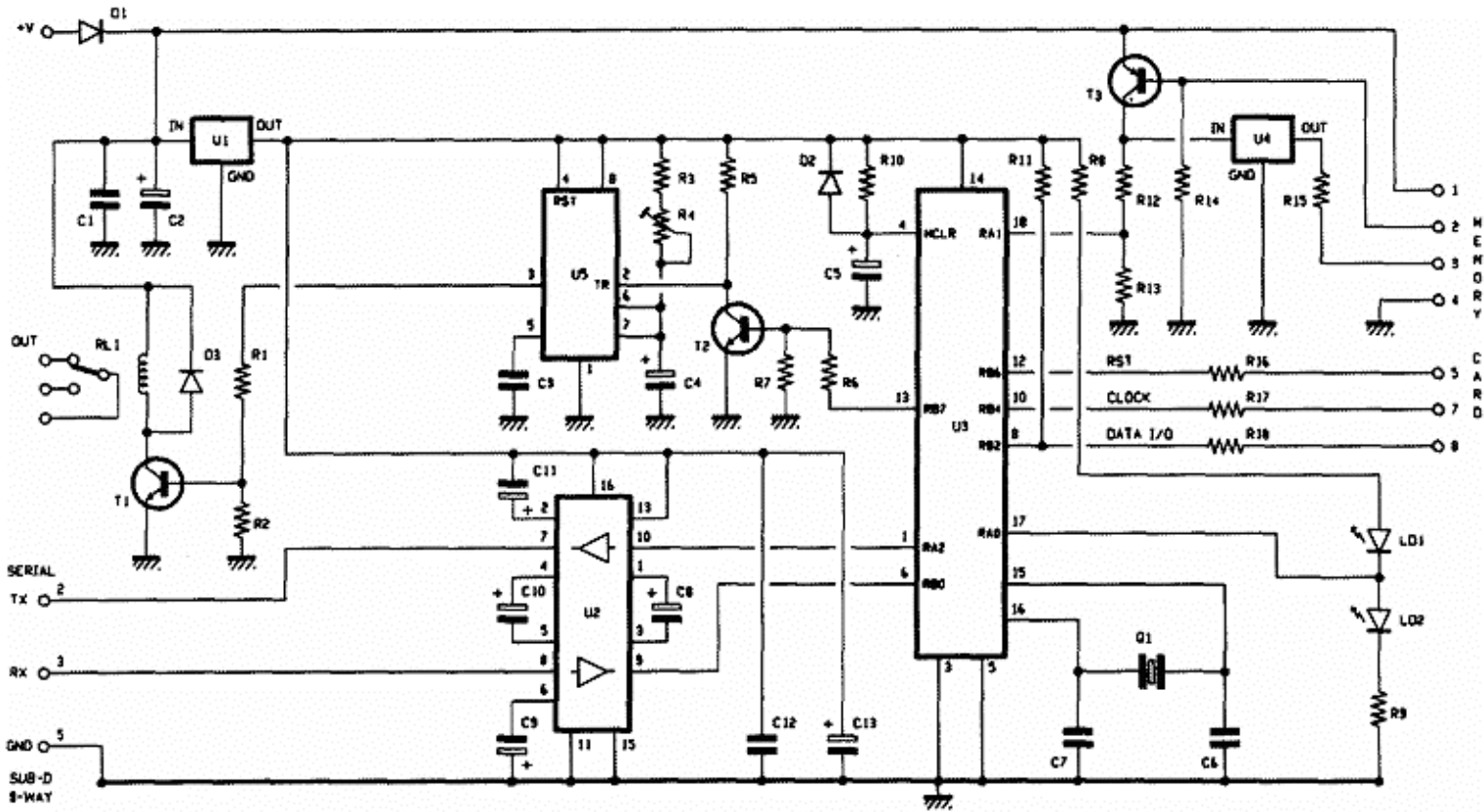
alcune informazioni relative all'autorità di certificazione che lo ha rilasciato, la firma digitale apposta dall'autorità di certificazione ed il periodo di validità.

Il paradigma è il seguente: un individuo compila una richiesta di certificazione con i suoi dati e la sua chiave pubblica e la invia ad una autorità di certificazione. Quest'ultima verifica l'autenticità dei dati e, se il responso è positivo, produce un certificato e lo rilascia al richiedente firmandolo con la propria chiave privata. Il richiedente può ora inviare il proprio certificato ad un altro individuo per farsi autenticare e per consegnargli la propria chiave pubblica.

La verifica dell'identità dell'individuo avviene mediante il controllo della firma apposta sul certificato dall'autorità di certificazione che, pertanto, mette a disposizione di tutti la propria chiave pubblica. In realtà mette a disposizione il proprio certificato auto-firmato, o firmato da un'altra autorità di certificazione

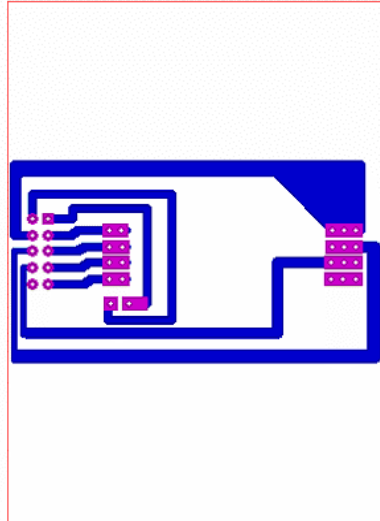
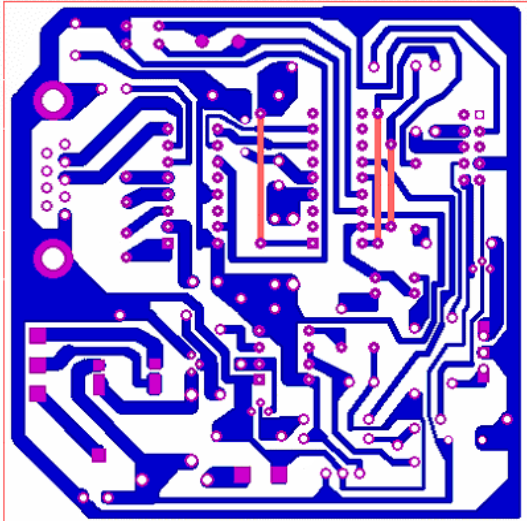
P A R T E P R A T I C A

SCHEMA ELETTRICO LETTORE SMART CARD:

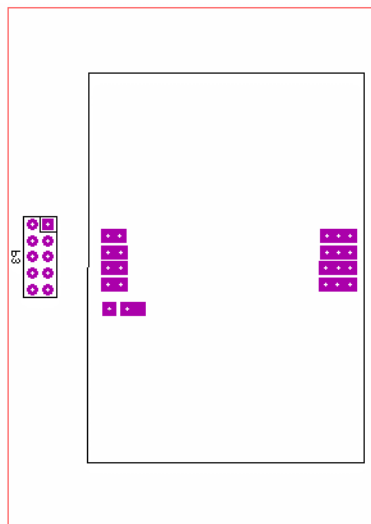
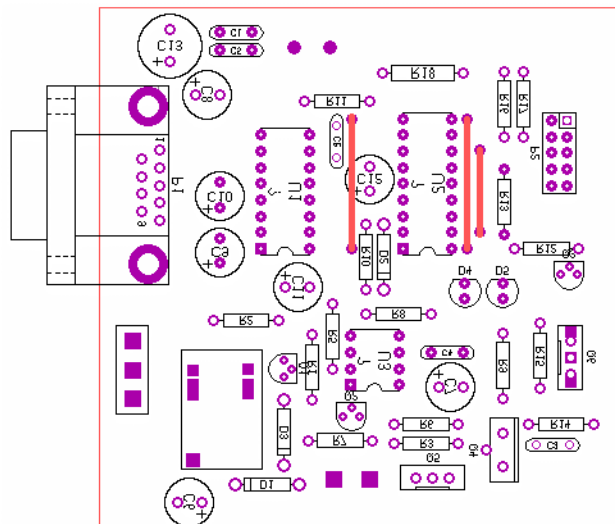


CIRCUITO SU BASETTA:

Lato Pise:



Lato componenti:



COMPONENTISTICA:

RESISTENZE:

R1: 15 K Ω

R2: 47 K Ω

R3: 47 K Ω

R4: Trimmer 470 K Ω miniatura MV

R5: 4,7 K Ω

R6: 47 K Ω

R7: 47 K Ω

R8: 470 Ω

R9: 470 Ω

CONDENSATORI:

C1: 100 nF multistrato

C2: 220 μ F 25VL elettrolitico rad.

C3: 100 nF multistrato

C4: 47 μ F 25VL elettrolitico rad.

C5: 10 μ F 25VL elettrolitico rad.

C6: 22 pF ceramico

C7: 22 pF ceramico

DIODI:

D1: Diodo 1N4004

D2: Diodo IN4148

D3: Diodo 1N4004

ALTRO:

RL1: Relè 12V min. 1 scambio

U1: LM7805

U2: MAX232

U3: PIC16F628

U4: 78L05

R10: 10 K Ω

R11: 10 K Ω

R12: 33 K Ω

R13: 33 K Ω

R14: 10 K Ω

R15: 47 K Ω

R16: 1 K Ω

R17: 1 K Ω

R18: 1 K Ω

C8: 10 μ F 25VL elettrolitico rad.

C9: 10 μ F 25VL elettrolitico rad.

C10: 10 μ F 25VL elettrolitico rad.

C11: 10 μ F 25VL elettrolitico rad.

C12: 100 nF multistrato

C13: 470 μ F 25VL elettrolitico rad.

LD1: Led verde 5 mm

LD2: Led rosso 5 mm

U5: NE555

T1: BC547B transistor NPN

T2: BC547B transistor NPN

T3: BC557B transistor PNP

Q1: Quarzo 4 Mhz

RELAZIONE:

Per la realizzazione di questa tesi ho adoperato differenti programmi tra i quali DELPHY5 per la realizzazione della parte software, CIRCAD98 per la realizzazione della basetta

riguardante la parte hardware, MPLab per la compilazione del programma in assembler, ICProg per programmazione del PIC16F628, ed infine Hyperterminal per verificare il funzionamento base del programmatore affinché permettesse di interagire tramite PC con la Smart Card.

Programma PIC16F628:

Di seguito è riportato il programma in assembler relativo al PIC16F628 con relative istruzioni per l'interazione con la smart card ISO 7818

```
Filename Smc4.asm
;
;
*****
*****
;
; Titolo: Lettore Smart Card ISO 7816
;
; Descrizione: Lettore Smart Card ISO 7816
;
; Microcontrollore utilizzato: PIC16F628
;
; Autore: Massimo Frassine
;
; Data ultimo aggiornamento: 02-05-2005

; Versione: 1.0

; !!! Osservazioni:

; Le linee usate sono;

; RA0 = RST
; RA1 = CLK
; RA2 = I/O
; RA3 = CARD_IN

; RB0 =
; RB1 = RX dal MAX232 pin (9 o 12) al MAX232 (8 o 13) viene dal pin 3 DB9 Femmina
; RB2 = TX al MAX232 pin (10 o 11) dal MAX232 (7 o 14) vanno al pin 2 DB9 Femmina
; RB3 =
; RB4 =
; RB5 =
; RB6 =
; RB7 =

; Elenco comandi
;
; prima lettera:
; W= Write
; R = Read
```

;
;
;
; seconda lettera:
;
; M= M ain
; S = S ecurity
; P = P rotect

; Esempi:

; RM senza altro legge tutta la memoria della SMART CARD a partire dall'indirizzo 0x00 fino a 0xFF (256 byte)
; RM40 senza altro legge la memoria della SMART CARD a partire dall'indirizzo 0x40 fino a 0xFF
; RM 6007 comando completo legge la memoria della SMART CARD a partire dall'indirizzo 0x60 fino a 0x67 (7 byte)

; RS comando completo legge i quattro byte che rappresentano il SECURITY CODE (nelle CARD nuove è = FF FF FF FF)
; RP comando completo legge i quattro byte che rappresentano (per ogni bit) lo stato di protezione (0) o non protezione (1) dei primi 32 byte della MEMORY-CARD

; WM 6047 comando completo scrive la memoria della SMART CARD all'indirizzo 0x60 con il valore 0x47

; CR segue il comando "R"eset & answer e restituisce i quattro byte del Manufactory code
; CVxyyyzz segue la procedura di VALIDAZIONE:
;
; 1) lettura numero tentativi (EC Error counter)
;
; 2) riscrittura numero tentativi / 2 (shift a dx) (es EC = 7 (111) scrivo EC = 3 (011))
;
; 3) confronto primo byte VD (Verification data)
;
; 4) confronto secondo byte VD (Verification data)
;
; 5) confronto terzo byte VD (Verification data)
;
; 6) ripristina il numero di tentativi (EC = 7)
;
; 7) legge il codice di sicurezza e lo invia al PC per la verifica

; ogni comando è sempre terminato dal Return (CR)

;

;

VERSIONE EQU 0 ; 0 = EEPROM 1 = Test Simulato

TEST EQU 0 ; 0 = NO TEST 1 = TEST

LIST p=16F628A ; Rif. Microcontrollore per compilatore
ERRORLEVEL -202,-302,-303,-301,-305,-306

IF VERSIONE == 0

__CONFIG 0x3F62 ; 0x3F22 B'11111100100010'

ENDIF

IF VERSIONE == 1

__CONFIG 0x3F22 ; 0x3F22 B'11111100100010'

ENDIF

;

; FILE di INCLUSIONE

;

#INCLUDE P16F628A.INC
#INCLUDE C:\DOCUMENTI\DOC_MPLAB\MACROS16.INC

;

RAM_BANK0 EQU 0x20
END_BANK0 EQU 0x7F

RAM_BANK1 EQU 0xA0
END_BANK1 EQU 0xFF

; DEFINIZIONE VARIABILI DI SERVIZIO

; VARIABILI GENERICHE COMUNI

;

CBLOCK RAM_BANK0

user : .4 ; Locazioni di uso generico.

gen_var : .4

cnt0

cnt1

c1

c2

c3

copy_c1

copy_c2

copy_c3

id1

id2

id3

i

d

dep_ascii
checksum : .2

num_sec
timer_dsec

;

; **VARIABILI DI TRASMISSIONE**

;

buff_tx_232
buff_rx_232 : .20
buff_comando : .10

p_rx_buff
p_buff_com

num_chr
user_232

dep_byte : .20

errore

;

; **VARIABILI DI TIPO FLAG**

;

flg_time ; Codice Flag.

;
; bit 0 =
; bit 1 =
; bit 2 =
; bit 3 =
; bit 4 =
; bit 5 =
; bit 6 =
; bit 7 =

flg_gen ; Codice Flag.

;
; bit 0 = RBIN
; bit 1 = RX_END
; bit 2 = ECHO_ON
; bit 3 =
; bit 4 =
; bit 5 =
; bit 6 =
; bit 7 =

;maschera
;num_par

;-----
; **VARIABILI DI INTERRUPT**
;-----

save_intcon **;** Mem. salvataggio Reg. INTCON
save_w **;** Mem. salvataggio Reg. W
save_stat **;** Mem. salvataggio Reg. STATUS
save_fsr **;** Mem. salvataggio Reg. FSR
save_pclath **;** Mem. salvataggio Reg. PCLATH

count_irq
user_irq
p_irq_rx_buff

ENDC

;-----
; **VARIABILI SOVRAPPOSTE**
;-----

;-----
; **VARIABILI IN PAG 1**
;-----

CBLOCK RAM_BANK1

;divisore **:** 2
;dividendo **:** 2
;resto **:** 2
;user_div **:** 2

;count_div

ENDC

;-----
; **DEFINIZIONE COSTANTI**
;-----

LONG_CODE **EQU .12**

;-----
; **COSTANTI per I/O**
;-----

DDRA **EQU B'11111100'**
DDRB **EQU B'11111111'**

DEF_PA EQU B'00000000'
DEF_PB EQU B'00000000'

MASK_FRONTE EQU B'01000000'

; I/O PORTA

SMART_RST EQU 0
SMART_CLK EQU 1
SMART_IO EQU 2
SMART_DET EQU 3

; I/O PORTB

MASK_IN EQU B'00000100'
MASK_OUT EQU B'11111011'

; Codici comandi PROTOCOLLO

STR_LUNG EQU .3

; Dichiarazione bit registro dei flag

RBIN EQU 0
RX_END EQU 1
ECHO_ON EQU 2

; COSTANTI per TIMER

FOSC EQU .4000000 ; Frequenza Clock di sistema

; Calcolo RTCC

; $256 - ((TON/PRESC)*fclock/4) = Tclk$
; $(256-Tclk)*(1/(fclock/4/PRESC)) = TON$

IF VERSIONE == 0

SCLK EQU .99 ; 99 Tick = 10 mSec con frequenza 4MHz
PRESCALER EQU .5 ; PRESCALER = 1/64
NUM_TICK EQU .10 ; Numero cicli di IRQ (10ms 4MHz) per ottenere 100mS

TIME_BIT EQU .165 ; 165 Tick = 200uS
BIT_PRESCALER EQU .0

ENDIF

IF VERSIONE == 1

SCLK EQU .200 ; Tick = 10 mSec con frequenza 4MHz
PRESCALER EQU .5 ; PRESCALER = 1/64
NUM_TICK EQU .1 ; Numero cicli di IRQ (10ms 4MHz) per ottenere 100mS

TIME_BIT EQU .200
BIT_PRESCALER EQU .0

ENDIF

;

;
BIT FLAG

;

;

;
Flg timer

;

TS EQU 0
TL1 EQU 1
TL2 EQU 2
TMS EQU 6

;

;
Flg code

;

CODE_ON EQU 0
TEMPO_TRANS EQU 1
RX_CODE EQU 2

TOUT EQU 6
SEGNO EQU 7

;

;
COSTANTI DI TRASMISSIONE

;

BAUD_RATE EQU .9600
BAUD_COUNT EQU .25

CR EQU .13
LF EQU .10
SP EQU .32
CTRL_Z EQU .1
ACK EQU .6
NAK EQU .15
RBIN EQU .0

;

;
codici comandi Smart Card

```
;-----  
CODECVD EQU 0x33 ;Compare Verification Data  
CODERMM EQU 0x30 ;Read Main Memory  
CODERSM EQU 0x31 ;Read Security Memory  
CODERPM EQU 0x34 ;Read Protection Memory  
CODEWMM EQU 0x38 ;Write Main Memory  
CODEWSM EQU 0x39 ;Write Security Memory  
CODEWPM EQU 0x3C ;Write Protection Memory
```

```
;-----  
; Codici protocollo di trasmissione  
;-----
```

```
MAIN_MEM EQU 'M'  
SECURITY_MEM EQU 'S'  
PROTECT_MEM EQU 'P'  
RESET_ATR EQU 'R'  
VALID_CARD EQU 'V'  
  
SCRIVI EQU 'W'  
LEGGI EQU 'R'  
COMANDO EQU 'C'
```

```
;-----  
; Indirizzi speciali della CARD  
;-----
```

```
FM_CARD EQU 0  
MM_CARD EQU .32
```

```
;-----  
; CODICI ERRORI  
;-----
```

```
OK EQU 0x00  
  
ERR_COMM EQU 0x0 ;EF  
ERR_TIPO EQU 0x1 ;EE  
ERR_WR EQU 0x2 ;EB  
ERR_NB EQU 0x3 ;FD  
ERR_NUM EQU 0x4 ;EA  
BLOCK_CARD EQU 0x5 ;E9  
ERR_CRC EQU 0x6 ;FE
```

```
;-----  
; DEFINIZIONI  
;-----
```

```
#DEFINE CARD_DETECT? btfss PORTA,SMART_DET  
  
#DEFINE DATA_ON? btfss PORTA,SMART_IO
```

```

#DEFINE DATA_OFF? btfsc PORTA,SMART_IO

#DEFINE CLK_ON          bsf  PORTA,SMART_CLK
#DEFINE CLK_OFF        bcf  PORTA,SMART_CLK
#DEFINE RST_ON          bsf  PORTA,SMART_RST
#DEFINE RST_OFF        bcf  PORTA,SMART_RST
#DEFINE DATA_ON       bsf  PORTA,SMART_IO
#DEFINE DATA_OFF      bcf  PORTA,SMART_IO

```

```

;-----
;          MACRO
;-----

```

```

_DATA_OUTmacro          ; Data = OUT
    banksel    TRISA
    movfw TRISA
    andlw MASK_OUT
    movwf TRISA
    banksel    PORTA
    endm

```

```

_DATA_IN macro          ; Data = IN
    banksel    TRISA
    movfw TRISA
    iorlw MASK_IN
    movwf TRISA
    banksel    PORTA
    endm

```

```

;-----
;          Vettori
;-----

```

```

;          VETTORE DI RESET ( POWER ON )

```

```

    ORG 0x0
    clrf PCLATH          ; PAGE0
    goto main

```

```

;          VETTORE DI INTERRUPT

```

```

    ORG 0x4
    goto irq_exec

```

```

;-----
;          Programma Principale
;-----

```

```

main      call  initgen
main0     call  test_232

```

```

main_loop call  automa          ; Esecuzione sequenza macchina a stati

```

goto main_loop

```
;  
-----  
;  
; FUNZIONE : INITGEN  
;  
; DESCRIZIONE : Inizializzazione del microcontrollore  
;  
; PAGE : 0  
;  
-----
```

; Inizializzazione RAM Bank0 e Bank1

```
initgen    movlw RAM_BANK0  
           movwf FSR  
init_0     clrf  INDF  
           movfw FSR  
           xorlw END_BANK0  
           btfsc STATUS,Z  
           goto  init_1           ; Clear BANK 0  
           incf  FSR  
           goto  init_0
```

```
init_1     movlw RAM_BANK1  
           movwf FSR  
init_10    clrf  INDF  
           movfw FSR  
           xorlw END_BANK1  
           btfsc STATUS,Z  
           goto  init_2           ; Clear BANK 0  
           incf  FSR  
           goto  init_10
```

; Inizializzazione linee I/O

```
init_2     movlw DEF_PA           ; Inizializza stato PORT-A e PORT-B  
           movwf PORTA  
           movlw DEF_PB  
           movwf PORTB  
  
           movlw 0x7  
           movwf CMCON  
  
           banksel    RAM_BANK1  
  
           movlw DDRA  
           movwf TRISA  
           movlw DDRB  
           movwf TRISB
```

; Inizializzazione TIMER TMR0

```

bcf  OPTION_REG,PSA ; Prescaler usato da TMR0
bcf  OPTION_REG,T0CS ; Sorgente di clock interna
movfw OPTION_REG
andlw 0xF8
iorlw PRESCALER ; Modifica valore prescaler
movwf OPTION_REG
;
bcf  OPTION_REG,NOT_RBPU ; Attiva resistori pull up sul PORTB
bsf  OPTION_REG,INTEDG ; Irq sul fronte di salita

```

```
banksel RAM_BANK0
```

```

movlw SCLK ; Preset durata TICK di sistema
movwf TMR0 ; Attiva TMR0.
movlw NUM_TICK
movwf count_irq ; Conta "n" cicli per fare 100mSec

```

; Inizializzazione TMR1

```

movlw B'00111000' ; TIMER1 on con clock interno Fin/4
movwf T1CON

```

; Inizializzazione USART

```

bcf  PIR1,TXIF ; Clear flag TX-IRQ
bcf  PIR1,RCIF ; Clear flag RX-IRQ

```

```
banksel RAM_BANK1
```

```

movlw BAUD_COUNT
movwf SPBRG ; set baud rate 9600

```

```

bcf  TXSTA,SYNC
bsf  TXSTA,BRGH
bsf  TXSTA,TXEN ; Abilita la trasmissione

```

```
bsf  PIE1,RCIE ; Set RX IRQ
```

```
banksel RAM_BANK0
```

```

bsf  RCSTA,SPEN ; Abilita USART RB2 = TX e RB1 = RX
bsf  RCSTA,CREN ; Abilita la ricezione

```

```

movlw buff_rx_232 ; reinizializza p_irq_rx_buff
movwf p_irq_rx_buff
movwf p_rx_buff

```

; Inizializzazione registro INTCON

```

clrf INTCON
bsf  INTCON,T0IE ; Set TMR0 IRQ ON.
bsf  INTCON,PEIE ; Start EXTERN IRQ

```

bsf INTCON,GIE ; Start IRQ generale.

return

```
;  
;  
; FUNZIONE : AUTOMA  
;  
; DESCRIZIONE : implementa l'automa (macchina a stati)  
; Input: stato attuale dell'automa  
; Output: stato prossimo dell'automa  
;  
; PAGE : 0  
;  
;
```

automa btfss flg_gen,RX_END ; Attende comando dal canale seriale

**goto automa
bcf flg_gen,RX_END**

call check_exec ; Verifica la correttezza del comando

xorlw OK

btfss STATUS,Z

goto automa_err

call exec_command ; Esegue il comando

return

automa_err call exec_err_7

goto automa

```
;
```

check_exec clrf num_chr

movlw buff_comando ; reinizializza buffer comando

movwfp_buff_com

check_0 movlw (buff_rx_232 + .20) ; Copia i caratteri del messaggio ricevuto nel BUFFER COMANDO

subwf p_rx_buff,W

btfss STATUS,C

goto check_1

movlw buff_rx_232 ; reinizializza p_rx_buff

movwfp_rx_buff

check_1 movwfp_rx_buff ; legge i caratteri ricevuti fino al CR e ...

movwfFSR

incf p_rx_buff,f

movfwINDF

movwfuser

xorlw CR

btfsc STATUS,Z

```

ricevuto      goto   check_2                ; passa ad intpretare ed eseguire il comando

movfw p_buff_com      ; ... li deposita nel buffer comando
movwf FSR
incf  p_buff_com,f
movfw user
movwf INDF
incf  num_chr        ; . conta i caratteri
goto  check_0

check_2      retlw  OK

check_err1   retlw  ERR_CRC

; -----

exec_command movlw buff_comando        ; reinizializza buffer comando
movwf p_buff_com

movfw p_buff_com
movwf FSR
movfw INDF        ; Analizza il messaggio (1° carattere)
xorlw SCRIVI      ; se è la lettera "W"rite...
btfsc STATUS,Z
goto  scrivi_card

movfw INDF
xorlw LEGGI      ; se è la lettera "R"ead
btfsc STATUS,Z
goto  leggi_card

movfw INDF
xorlw COMANDO    ; se è la lettera "C"ommand
btfsc STATUS,Z
goto  esegui_comando
goto  exec_err_1

; -----

scrivi_card  incf  FSR
movfw INDF        ; Analizza il messaggio (2° carattere)
xorlw MAIN_MEM    ; se è la lettera "M"ain
btfsc STATUS,Z
goto  scrivi_card1

movfw INDF        ; Analizza il messaggio (2° carattere)
xorlw SECURITY_MEM ; se è la lettera "S"ecurity
btfsc STATUS,Z
goto  scrivi_card2

movfw INDF        ; Analizza il messaggio (2° carattere)

```

```
xorlw PROTECT_MEM ; se è la lettera "P" rotect
btfsc STATUS,Z
goto scrivi_card3
goto exec_err_2
```

```
scrivi_card1 movlw CODEWMM
goto scrivi_com1
```

```
scrivi_card2 movlw CODEWSM
goto scrivi_com1
```

```
scrivi_card3 movlw CODEWPM
```

```
scrivi_com1 movwf c1
```

```
movfw num_chr
sublw .2
btfss STATUS,C
goto scrivi_c1
movlw MM_CARD
movwf c2
movlw 0xFF
movwf c3
goto scrivi_com
```

```
scrivi_c1 incf FSR
movfw INDF ; Unisce i successivi due caratteri ...
call ascii_hex
movwf user
incf FSR
movfw INDF
call ascii_hex
swapf user
iorwf user,W
movwf c2 ; ...in un unico byte (INDIRIZZO)
```

```
movfw num_chr
sublw .4
btfss STATUS,C
goto scrivi_c2
movlw 0xFF
movwf c3
goto scrivi_com
```

```
scrivi_c2 incf FSR
movfw INDF ; Unisce i successivi due caratteri ...
call ascii_hex
movwf user
incf FSR
movfw INDF
call ascii_hex
swapf user
```

```

iorwf user,W
movwfc3          ; ; ...in un unico byte (DATO)

scrivi_com      call  tx_command
;              goto  exec_ret

               call  command          ; invia il comando alla CARD
               call  processing       ; attende che sia eseguito
               xorlw OK
               btfss STATUS,Z
               goto  exec_err_3

               goto  exec_ret

; -----

leggi_card      incf  FSR
               movfw INDF             ; Analizza il messaggio (2° carattere)
               xorlw MAIN_MEM        ; se è la lettera "M"ain
               btfsc STATUS,Z
               goto  leggi_card1

               movfw INDF             ; Analizza il messaggio (2° carattere)
               xorlw SECURITY_MEM     ; se è la lettera "S"ecurity
               btfsc STATUS,Z
               goto  leggi_card2

               movfw INDF             ; Analizza il messaggio (2° carattere)
               xorlw PROTECT_MEM     ; se è la lettera "P"rotect
               btfsc STATUS,Z
               goto  leggi_card3
               goto  exec_err_2

leggi_card1     movlw CODERMM
               goto  leggi_com2

leggi_card2     movlw CODERSM
               goto  leggi_com1

leggi_card3     movlw CODERPM

leggi_com1      movwfc1
               clrf  c2
               clrf  c3
               movlw .4
               movwf num_chr
               goto  leggi_com

leggi_com2      movwfc1

               movfw num_chr
               sublw .2

```

```

btfss STATUS,C
goto leggi_c1
movlw FM_CARD
movwf c2
movlw 0xFF
movwf num_chr ; N° caratteri da leggere = 255
goto leggi_com

```

```

leggi_c1      incf FSR
movfw INDF   ; Unisce i successivi due caratteri ...
call  ascii_hex
movwf user
incf FSR
movfw INDF
call  ascii_hex
swapf user
iorwf user,W
movwf c2    ; ...in un unico byte (INDIRIZZO)

```

```

movfw num_chr
sublw .4
btfss STATUS,C
goto leggi_c2
movfw c2
sublw 0xFF
movwf num_chr ; N° caratteri da leggere = 256-indirizzo iniziale
incf num_chr
goto leggi_com

```

```

leggi_c2      incf FSR
movfw INDF   ; Unisce i successivi due caratteri ...
call  ascii_hex
movwf user
incf FSR
movfw INDF
call  ascii_hex
swapf user
iorwf user,W
movwf num_chr ; ...in un unico byte (N° CARATTERI da LEGGERE)

```

```

movfw c2
addwf num_chr,W
btfsc STATUS,C
goto  exec_err_4

```

```

leggi_com    clrf c3 ; 3° carattere ininfluente
call tx_command

call read_tx ; legge e trasmette
goto exec_ret

```

```

; -----

```

```

esegui_comando   incf   FSR
                   movfw INDF           ; Analizza il messaggio (2° carattere)
                   xorlw RESET_ATR       ; se è la lettera "R" eset & answer to reset
                   btfsc STATUS,Z
                   goto   esegui_res_atr

```

```

                   movfw INDF           ; Analizza il messaggio (2° carattere)
                   xorlw VALID_CARD       ; se è la lettera "V" alidazione
                   btfsc STATUS,Z
                   goto   esegui_v_card
                   goto   exec_err_2

```

```

esegui_res_atr   call   answer_to_res
                   movlw .4
                   call   tx_n_byte
                   goto   exec_ret

```

```

esegui_v_card   call   validazione
                   goto   exec_ret

```

; -----

```

validazione   movfw num_chr
                   xorlw .8
                   btfss STATUS,Z
                   goto   exec_err_5

```

```

                   incf   FSR
                   movfw INDF
                   call   ascii_hex
                   movwf user
                   incf   FSR
                   movfw INDF
                   call   ascii_hex
                   swapf user
                   iorwf user,W
                   movwf id1           ; converte ASCII in HEX (ID1)

```

```

                   incf   FSR
                   movfw INDF
                   call   ascii_hex
                   movwf user
                   incf   FSR
                   movfw INDF
                   call   ascii_hex
                   swapf user
                   iorwf user,W
                   movwf id2           ; converte ASCII in HEX (ID2)

```

```

                   incf   FSR
                   movfw INDF

```

```
call  ascii_hex
movwf user
incf  FSR
movfw INDF
call  ascii_hex
swapf user
iorwf user,W
movwfid3 ; converte ASCII in HEX (ID3)
```

```
movlw CODERSM
movwfc1
clrf  c2
clrf  c3
```

```
movlw .4
movwf num_chr
```

```
call  leggi_com ; Visualizza stato contatore TENTATIVI
```

```
movfwdep_byte
btfsc STATUS,Z
goto  exec_err_6 ; Stop validazione
```

```
;  
;  
movlw 2  
call  Tnsec
```

```
movlw CODEWSM
movwfc1
clrf  c2
bcf  STATUS,C
rrf  dep_byte
movfwdep_byte
movwfc3
```

```
call  scrivi_com ; Riduce di uno il suddetto contatore
```

```
;  
;  
movlw 2  
call  Tnsec
```

```
movlw CODECVD
movwfc1
movlw 1
movwfc2
movwid1
movwfc3
```

```
call  scrivi_com ; Invia il primo ID
```

```
;  
;  
movlw 2  
call  Tnsec
```

```
movlw CODECVD
```

```

movwf c1
movlw 2
movwf c2
movfwid2
movwf c3

call scrivi_com          ; Invia il secondo ID

;
;
movlw 2
call Tnsec

movlw CODECVD
movwf c1
movlw 3
movwf c2
movfwid3
movwf c3

call scrivi_com          ; Invia il terzo ID

;
;
movlw 2
call Tnsec

movlw CODEWSM
movwf c1
clrf c2
movlw 7
movwf c3

call scrivi_com          ; Ripristina il contatore di TENTATIVI

;
;
movlw 2
call Tnsec

movlw CODERSM
movwf c1
clrf c2
clrf c3

movlw .4
movwf num_chr

call leggi_com           ; Visualizza stato contatore TENTATIVI

;
;
return

;-----
exec_ret    return

```

; -----

```
exec_err_1  movlw ERR_C0MM      ; Comando <> R W C
            goto  exec_err

exec_err_2  movlw ERR_TIPO      ; Tipo <> M S P R
            goto  exec_err

exec_err_3  movlw ERR_WR        ; Errore in WRITE
            goto  exec_err

exec_err_4  movlw ERR_NB        ; N° byte errato
            goto  exec_err

exec_err_5  movlw ERR_NUM       ; N° carattereri RX non conforme
            goto  exec_err

exec_err_6  movfw BLOCK_CARD
            goto  exec_err

exec_err_7  movfw ERR_CRC
            goto  exec_err

exec_err    movfw user
            movlw NAK
            call  tx_chr

loop2       movfw user
            call  vis_messaggio

            call  tx_cr
            return
```

; -----

```
vis_messaggio  movfw user          ; N° messaggio
               clrf  user+1
               clrf  user+2
               movfw user
               btfsc STATUS,Z      ; salta se <> da 0
               goto  vis_ok

vis_loop       movfw user+1
               call  load_chr
               incf  user+1
               xorlw 0
               btfss STATUS,Z
               goto  vis_loop
               decfsz user
               goto  vis_loop
```

vis_ok

```
vis_loop0    movfw user+1
             call   load_chr
             xorlw  0
             btfsc STATUS,Z
             goto  vis_m_ret
             call   tx_chr
             incf  user+1
             goto  vis_loop0
```

vis_m_ret return

```
load_chr     movlw ELENCO_MESS >> 8
             movwf PCLATH
             movlw ELENCO_MESS & 0xFF
             addwf user+1,W
             btfsc STATUS,C
             incf  PCLATH
             movwf PCL
```

```
ELENCO_MESS  dt "ALL. 1", 0
             dt "ALL. 2", 0
             dt "ALL. 3", 0
             dt "ALL. 4", 0
             dt "ALL. 5", 0
             dt "ALL. 6", 0
             dt "ALL. 7", 0
             dt "ALL. 8", 0
```

; -----

```
read_tx      movfw num_chr
             movwf user+2           ; salva N° caratteri da leggere
```

```
read_tx0     call   copy_par_c     ; copia i parametri del comando per una eventuale
ripetizione
```

```
             call   command
```

```
             movfw user+2
             sublw .16
             btfsc STATUS,C
             goto  read_tx1
             movlw .16             ; li scompone in gruppi da 16 chr
             movwf num_chr
             goto  read_tx_loop
```

```
read_tx1     movfw user+2
             movwf num_chr
```

```
read_tx_loop call   read_n_byte     ; read_memory (1 .. 16 byte)
```

```
movfw copy_c1
xorlw CODERPM
btfss STATUS,Z
goto read_tx10
```

```
call swap_dati
```

```
read_tx10 movfw num_chr
call tx_n_byte ; trasmette al PC i byte appena letti
```

```
movlw .16
subwf user+2,f
btfss STATUS,C
goto read_tx_fine
btfsc STATUS,Z
goto read_tx_fine
```

```
movfw user+2
xorlw .239
btfss STATUS,Z
goto read_tx2
```

```
movlw .240
movwf user+2
```

```
read_tx2 movfw copy_c1
movwf c1
movfw copy_c2
addlw .16
movwf c2
clrf c3
goto read_tx0
```

```
read_tx_fine return
```

; -----

```
swap_dati movlw dep_byte
movwf FSR
movlw .4
movwf gen_var
sw0 movlw .8
movwf gen_var+1
clrf gen_var+2
sw1 rrf INDF
rlf gen_var+2
decfsz gen_var+1
goto sw1
movfw gen_var+2
movwf INDF
incf FSR
```

```
    decfsz gen_var
    goto sw0
```

```
    return
```

```
; -----
```

```
tx_n_byte    movwf user
              movlw dep_byte
              movwf FSR
```

```
tx_n_loop    movfw INDF
              call tx_hex
              movlw SP
              call tx_chr
```

```
    incf FSR
    decfsz user
    goto tx_n_loop
```

```
    call tx_cr
```

```
    return
```

```
; -----
```

```
ascii_hex    movwf dep_ascii
              sublw '9'
              btfss STATUS,C
              goto ascii_lettera
              movfw dep_ascii
              andlw 0x0F
              goto ascii_ret
```

```
ascii_lettera movlw .55
              subwf dep_ascii,W
```

```
ascii_ret    return
```

```
; -----
```

```
copy_par_c   movfw c1
              movwf copy_c1
              movfw c2
              movwf copy_c2
              movfw c3
              movwf copy_c3
              return
```

```
; -----
```

```
; Primitive SMART-CARD
```

```
; -----
```

```
; Fornisce l'impulso di start alla carta
```

start

```
DATA_ON
nop
CLK_ON
call delay_5u
DATA_OFF
call delay_5u
CLK_OFF
return
```

;

;
Fornisce l'impulso di stop alla trasmissione

stop

```
DATA_OFF
call delay_5u
CLK_ON
call delay_5u
DATA_ON
call delay_5u
return
```

;

;
Fornisce l'impulso di break alla trasmissione

brk

```
CLK_OFF
nop
RST_ON
call delay_10u
RST_OFF
return
```

;

;
Comunica alla carta i dati necessari all'esecuzione del comando

;
Input: c1 (codice comando) - c2 (indirizzo) - c3 (dato)

;
Output: -

;

;

command

```
_DATA_OUT
nop
```

```
call start
```

```
call delay_5u
```

```
movlw .24
```

;
tx 24 bit COMANDO,INDIRIZZO,DATO LSB --> MSB

```
movwfi
```

comm_loop bcf STATUS,C

```

    rrf    c3           ; DATO
    rrf    c2           ; INDIRIZZO
    rrf    c1           ; COMANDO
    btfss  STATUS,C
    goto  $+3
    DATA_ON
    goto  $+2
    DATA_OFF
    nop
    nop
    CLK_ON
    call  delay_10u
    CLK_OFF
    call  delay_10u
    decfsz i
    goto  comm_loop

    call  stop

    return

```

```

;-----
;
;   Legge un DATO dalla SMART_CARD e termina con un BRK
;   Input:
;   Output:          dato letto, LSB --> MSB
;-----

```

outgoing

```

read_memory    _DATA_IN
               call  delay_5u
               CLK_OFF           ; CLK OFF

               movlw .8           ; Legge un DATO dalla MEMORY-CARD LSB --> MSB
               movwfi

read_m_loop    call  delay_10u
               CLK_ON           ; CLK on
               call  delay_10u
               DATA_ON?
               goto  read_m_0
               bsf   STATUS,C
               goto  $+2
read_m_0       bcf   STATUS,C
               CLK_OFF           ; CLK off
               rrf   d
               decfsz i
               goto  read_m_loop

               call  brk

```

```
movfwd
return
```

```
;-----
```

```
read_n_byte movwfuser
movlw dep_byte
movwfFSR
```

```
_DATA_IN
```

```
CLK_OFF
call delay_5u
```

```
read_n0 movlw .8 ; Legge l' ID della MEMORY-CARD LSB --> MSB
movwfi
```

```
read_n_loop call delay_10u ; CLK on
CLK_ON
call delay_10u
DATA_ON?
```

```
goto read_n1
bsf STATUS,C
goto $+2
```

```
read_n1 bcf STATUS,C ; CLK off
CLK_OFF
rrf d
decfsz i
goto read_n_loop
```

```
movfwd
movwfINDF
incf FSR
```

```
decfsz user
goto read_n0
```

```
call brk
```

```
return
```

```
;-----
```

```
;
```

```
;
```

```
;
```

```
;
```

```
;
```

```
;
```

```
;
```

```
;-----
```

```
processing clrf i ; Start processing di ERASE/WRITE
call start
```

_DATA_IN

```
process_0    call  delay_10u
              CLK_ON
              call  delay_10u
              CLK_OFF
```

```
DATA_OFF?
goto  process_1
incf  i
goto  process_0
```

```
process_1    ;_DATA_OUT
```

```
movfwi
sublw 2
btfsc STATUS,C
retlw ERR_WR           ; operazione fallita
retlw OK              ; operazione riuscita
```

```
-----
;
;
;   Resetta la memoria e legge i primi 4 byte della Main Memory
;   Input:
;   Output:
;
;
;-----
```

```
answer_to_res    RST_OFF           ; start
                  CLK_OFF
```

```
call  delay_40u           ; Power-On Reset Time 100us
call  delay_40u
call  delay_10u
call  delay_10u
```

```
RST_ON           ; RST on
call  delay_5u
CLK_ON           ; CLK on
call  delay_10u
CLK_OFF          ; CLK off
call  delay_5u
RST_OFF         ; RST off
```

```
movlw .4
movwf num_chr

call  read_n_byte

;_DATA_OUT
```


; -----

```
test_RS232    movlw 0           ; Invia il set di caratteri
              movwf buff_tx_232
              movwf num_chr
```

```
tx_data      clrwdt
              movfw buff_tx_232
              call tx_chr
              incf buff_tx_232
              decfsz num_chr
              goto tx_data

              call tx_cr

              return
```

; -----

```
Tnsec        movwf num_sec
tns0         call T1sec
            decfsz num_sec
            goto tns0
            return
```

; -----

```
T1sec        call T05sec
T05sec

            IF VERSIONE == 0

            bcf INTCON,T0IF

            movlw .5           ; 5 = 500 mSec con tick = 100 mSec
            movwf timer_dsec
            bsf flg_time,TMS; Clear flg TMS

T1_loop      btfsz flg_time,TMS; Timeout on Timer_1G ?
            goto T1_loop      ; ...se si salta...

            ENDIF

            return
```

; -----

```
;
;
; FUNZIONE      : Ritardo di n msec a 4MHz ()
;
; DESCRIZIONE   : Ritardo di n mS
;
; PAGE         : 0
```

```

;
; -----
Tn_msec      movwf cnt1
wait_ms1     movlw .200          ; 200x5us = 1000 us = 1 ms
              movwf cnt0        ; LOAD LOOP COUNTER
wait_ms2     nop                ; [1] WAIT A WHILE
              nop                ; [1] WAIT A WHILE
              decfsz cnt0,f      ; [1] ALL INNER LOOPS DONE
              goto  wait_ms2     ; [2] ... NO, THEN DO NEXT
              clrwdt             ; SUB TOTAL = 5 CYCLES
              decfsz cnt1,f      ; [1] ALL OUTER LOOPS DONE
              goto  wait_ms1     ; [2] ... NO, THEN DO NEXT LOOP
              return

```

```

; -----
;
; FUNZIONE      : wait_us ()
;
; DESCRIZIONE   : Ritardo di n x 10uS
;
; PAGE          : 0
;
; -----

```

```

wait_10us     movwf cnt1
wait_us1      nop
              nop
              decfsz cnt1
              goto  wait_us1
              return

```

```

; -----
;
; Test canale seriale con invio del set completo ASCII
; -----

```

```

test_232      crlf  user
              crlf  num_chr
test_loop     movfw user
              movwf buff_tx_232
              call  tx_chr
              incf  user
              decfsz num_chr
              goto  test_loop
              return

```

```

; -----
;
; Primitiva trasmissione seriale RS232
; -----

```

```

tx_cr         movlw CR
              goto  tx_chr

```

```
tx_crlf    movlw CR
           call  tx_chr
           movlw LF
           goto  tx_chr
```

;-----

```
tx_hex_RBI bcf    flg_gen,RBIN
tx_hex     movwf buff_tx_232
           swapf buff_tx_232,W
           call  tx_hex_asc
           movfw buff_tx_232
           goto  tx_hex_asc
```

;-----

```
tx_hex_asc andlw 0x0F
           movfw user_232
           sublw 0x9
           btfsc STATUS,C
           goto  tx_ha0
           movfw user_232
           addlw 7
           goto  $+2
```

```
tx_ha0    movfw user_232
           addlw 0x30
           movfw user_232
           btfss flg_gen,RBIN
           goto  tx_chr
           movfw user_232
           xorlw '0'
           btfsc STATUS,Z
           return
           bcf    flg_gen,RBIN
           movfw user_232
```

;-----

```
tx_chr    bcf    INTCON,GIE

           banksel    RAM_BANK1
```

```
tx_loop0  btfss  TXSTA,TRMT
           goto  tx_loop0
```

```
           banksel    RAM_BANK0
```

```
           movfw TXREG           ; Deposita carattere da TX nel buffer di TX
```

```
           bsf    INTCON,GIE
```

```
           clrwdt
```


irq_msec goto irq_ret

;

; Operazioni eseguite se si riceve un carattere

;

irq_RX232 bcf PIR1,RCIF
movlw (buff_rx_232 + .20)
subwf p_irq_rx_buff,W
btfss STATUS,C
goto irq_RX0
movlw buff_rx_232 ; reinizializza p_irq_rx_buff
movwfp_irq_rx_buff

irq_RX0 movfw p_irq_rx_buff
movwf FSR ; aggiorna indirizzo buffer di RX
movfw RCREG
movwf user_irq
xorlw LF ; Abort LF
btfsc STATUS,Z
goto irq_ret
movfw user_irq

btfsc flg_gen,ECHO_ON
movwf TXREG ; ... e trasmette echo

movwf INDF ; Salva il dato ricevuto
incf p_irq_rx_buff
xorlw CR ; CR = fine messaggio
btfss STATUS,Z ; Se è il carattere ESC termina ...
goto irq_ret
bsf flg_gen,RX_END ; ... attivando il flag
goto irq_ret

;

irq_ret clrwdt
movfw save_pclath
movwf PCLATH
movfw save_fsr
movwf FSR
swapf save_stat,W
movwf STATUS
swapf save_w,f
swapf save_w,w

retfie ; Return from IRQ.

;

;

;

```

tx_echo          bcf  INTCON,GIE

                call  tx_cr

tx_echo_0        movlw (buff_rx_232 + .20) ; Copia i caratteri del messaggio ricevuto nel BUFFER
COMANDO
                subwf p_rx_buff,W
                btfss STATUS,C
                goto  tx_echo_1
                movlw buff_rx_232          ; reinizializza p_rx_buff
                movwfp_rx_buff

tx_echo_1        movfw p_rx_buff
                movwf FSR
                movfw INDF
                movwf user
                xorlw CR                   ; CR = fine messaggio
                btfsc STATUS,Z
                goto  tx_echo_ret
                movfw user
                call  tx_chr               ; Tx echo
                incf  p_rx_buff
                goto  tx_echo_0

tx_echo_ret      incf  p_rx_buff
                call  tx_cr

                bsf  INTCON,GIE
                return
; -----

tx_command       bcf  INTCON,GIE

                movlw ACK
                call  tx_chr
                movlw SP
                call  tx_chr
                movfw c1
                call  tx_hex
                movfw c2
                call  tx_hex
                movfw c3
                call  tx_hex
                call  tx_cr
                return
; -----

                END

```

Borland Delphi 5:

Delphi nasce come evoluzione del "Borland Turbo Pascal".

Il **Pascal** è un linguaggio ad alto livello che è stato sviluppato alla fine degli anni sessanta dal professor Niklaus Wirth a Zurigo.

Il suo scopo era quello di realizzare un linguaggio comprendente un piccolo numero di concetti fondamentali di programmazione sufficienti ad insegnarla in forma di disciplina logica e sistematica; nello stesso tempo voleva un linguaggio che fosse facile ed efficiente da implementare sulla maggior parte dei calcolatori.


Il suo successo nel conseguire questo obiettivo può essere misurato dalla rapida diffusione dell'uso del PASCAL sia come linguaggio usato nella didattica dei principi di programmazione (si usa tutt'oggi nelle università), sia come linguaggio effettivo per scrivere software di base ed applicativo.





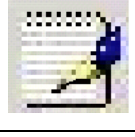




Le versioni di Delphi al momento disponibili sul mercato si dividono in 3 categorie :


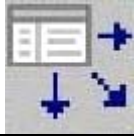





- La versione "**Standard**" come dice il nome stesso, è il livello base di Delphi per la realizzazione di applicazioni semplici che non richiedono l'utilizzo di database e l'accesso ad Internet, perlopiù indirizzate a studenti, hobbisti, principianti.
- La versione "**Developer**" o "**Professional**" è rivolta a coloro che utilizzano Delphi per lavoro e che necessitano di supporto per database e internet.
- Le "**Enterprise**", come dice il nome, sono rivolte alle aziende che necessitano del supporto per il controllo di versione, accesso a database server, internet, internazionalizzazione del software.

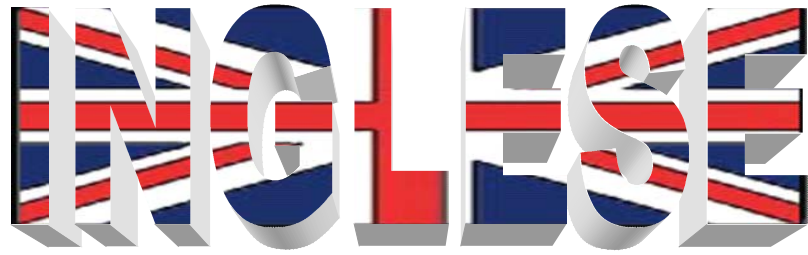
Il Delphi è un linguaggio di programmazione a oggetti orientati "object oriented" (cioè orientato agli oggetti) quindi la stesura del codice sorgente non è l'unica operazione da fare ma è accompagnata da un'impostazione grafica composta da finestre, bottoni, caselle di testo e molti altri oggetti.

Qui di seguito sono riportati in una tabella la maggior parte degli oggetti che abbiamo utilizzato per realizzare i nostri software

Oggetti Delphi :	Descrizione:
Form 	Componente che fornisce il supporto base del programma in cui vengono depositati gli oggetti. È un interfaccia grafica tipica di ogni applicazione basata su sistemi operativi di windows. Essa ha funzione di permettere all'utente di interagire con l'applicativo e quindi impartirgli degli "ordini" tramite l'uso di oggetti "visuali" che il programmatore include al suo interno allo scopo di soddisfare le specifiche del programma.

<p>Label</p> 	<p>La label è un'etichetta che l'utente non può selezionare o maneggiare, quali, titoli di testo o etichette di controllo che hanno scopo indicativo.</p>
<p>Edit</p> 	<p>Visualizza una zona di scrittura in cui l'utente può inserire o modificare una singola linea di testo. Con questo oggetto il programmatore può inserire stringhe di testo allo scopo di fornire dei dati al programma.</p>
<p>Button</p> 	<p>Il bottone è uno degli oggetti standard che permette di interagire con i programmi. La sua funzione principale è quella di far eseguire determinate operazioni ogni volta che viene premuto, le operazioni vengono stabilite dal programmatore tramite la scrittura di istruzioni particolari che vengono eseguite in sequenza.</p>
<p>Memo</p> 	<p>La memo è un oggetto che ha le stesse funzioni della EDIT con la differenza che il testo immesso può essere distribuito su più righe. Per esempio lo si utilizza per l'inserimento di commenti o appunti.</p>
<p>Log File</p> 	<p>È un oggetto non visuale che permette di registrare in un file di testo gli eventi più importanti di quel specifico programma. Con questo riusciamo a risalire ad eventuali problemi o solamente ad avere un resoconto del lavoro svolto.</p>
<p>Timer</p> 	<p>Il timer è un componente non visuale che serve a generare degli eventi dopo un tempo stabilito dal programmatore. La durata del periodo è specificata nella proprietà <i>interval</i>, espressa in millisecondi.</p>
<p>Check Box</p> 	<p>La Check Box presenta un'opzione che l'utente può scegliere tra sì/no oppure vero/falso. Utilizza le caselle di controllo per visualizzare un gruppo di scelta che non si esclude a vicenda. Gli utenti possono selezionare più di una casella di controllo in un gruppo.</p>
<p>Radio_Button</p> 	<p>1 La radio Button è un oggetto grafico simile alla Check Box soltanto che a differenza della prima, questa consente di selezionare solo una delle varie opzioni.</p>
<p>ADO Connection</p> 	<p>L'ADO Connection è uno dei componenti più importanti di Delphi 5; questo permette di comunicare con il Database fisico tramite il collegamento con l'ADO Query.</p>

<p>ADO Query</p> 	<p>È un componente che utilizza le istruzioni SQL per richiamare dei dati da una tabella fisica del database, e consente ai componenti “<i>data-aware</i>” di essere manipolati connettendosi con un componente <i>DataSource</i>.</p>
<p>Data Source</p> 	<p>Agisce come interconnessione tra i componenti dataset (come le tabelle) e Data-Aware (la griglia dove appaiono i dati che provengono da una tabella oppure da una query).</p>
<p>VaComm</p> 	<p>Componente che fornisce accesso alle porte seriali attraverso il sistema operativo Win 95/98 e NT. La comunicazione guidata a evento, è un metodo molto efficace per la manipolazione o interazione della porta seriale. In molte situazioni si può voler notificare il momento in cui avviene un evento, come quando un carattere arriva, o un cambiamento si presenta durante la rivelazione della portante.</p>
<p>TNMSNTP</p> 	<p>Questo componente abilita i controlli ActiveX che danno la possibilità alle applicazioni di comunicare via internet tramite l’utilizzo delle e-mail con i server SMTP . (ActiveX control that gives applications access to SMTP mail servers and mail posting capabilities).</p>
<p>DBGrid</p> 	<p>Il DatabaseGrid è una griglia che suddivide i dati di una tabella per permettere la visualizzazione in maniera ordinata, tabulare, simile ad un foglio elettronico. Questo componente fa un vasto uso delle proprietà campo Tfield per determinare visibilità della colonna, formato di affissione, ordinanti e così via.</p>
<p>hhALed</p> 	<p>È un oggetto che simula il funzionamento di un diodo led.</p>
<p>NmPop3</p> 	<p>È un componente invisibile che attiva i controlli activex e richiama la posta da UNIX o da altri server che utilizzano il protocollo POP3.</p>



HISTORY

Smart cards were invented and patented in [France](#) by [Roland Moreno](#) in the [1970s](#). Their first mass usage was payment in the French [payphones](#) starting from [1983](#) ([Télécarte](#)). The second one was the integration of a microchip into all French [debit cards](#) ([Carte Bleue](#)).

CONTACT SMARTCARD

The **ISO/IEC 7816** series of standards define:

- the physical shape of the smart card
- the positions and shapes of its electrical connectors
- the [communications protocols](#) and power voltages to be applied to those connectors
- the functionality
- the format of the commands sent to the card and the response returned by the card

The cards do not contain a [battery](#); power is supplied by the card reader.

In a *contact-type* smart card, the chip can be recognised by an area of gold-plated contacts about 1 cm² close to the short side of the card. Normally the contact communication is relatively slow (9.6-115.2 kbit/s). There is currently a trend towards implementing [USB 1](#) on these contacts (up to 10 Mbit/s), but there is not yet a final standard.

ISO 7816:5 defines numbering for ISO 7816 smart cards. An application identifier (AID) consists of an Registered Application Provider Identifier (RID), identifying the vendor, then a Proprietary Application Identifier Extension (PIX), identifying the application offered by the vendor. A RID can be either assigned by the ISO/IEC 7816-5 Registration Authority (TDC Services A/S), or be an [ISO 7812](#) IIN followed by FF hexadecimal.

CONTACTLESS SMARTCARD

A second type is the *non-contact type* called *contactless* smart card, where the chip communicates with the card reader through wireless self-powered induction technology (106-424kbits/s).

The standards for the [contactless](#) protocol for smart cards are **ISO/IEC 14443** (type A and B) from the year 2001. There have been proposals for ISO 14443 type C, D, E and F that have yet to be accepted by the ISO standards committee. An alternative standard for contactless smartcard is **ISO 15693**.

An example of a widely used [contactless](#) smartcard is Hong Kong's [Octopus card](#), which predates the ISO/IEC 14443 standard. For use on [public transportation](#), [Malaysia](#) introduced the [Touch 'n Go](#) smartcard in 1997, [Paris](#) introduced the [Calypso card](#) in October 2001, and [London](#) introduced the [Oyster card](#) in January 2004.

A related contactless technology is **RFID** (radio frequency identification) that in certain cases can be used for similar applications to contactless smartcard such as for **electronic toll collection**. RFID generally do not include writeable memory or microcontroller processing capability as contactless smartcard do.

There are dual-interface cards that implement contactless and contact interfaces on a single card with some shared storage and processing. An example is **Malaysia's** multi application smartcard identification called **MyKad** that uses both contact **Proton** and contactless **Mifare** (ISO 14443A) chips.

APPLICATIONS

The applications of smartcards include their use as credit or **ATM** cards, **SIMs** for mobile phones, authorization cards for pay television, high security identification and access control cards, **public transport** tickets, etc.

Smart cards may also be used as **electronic wallets**. The smart card chip can be loaded with electronic money, which can be used to pay parking meters, vending machines, and merchants. **Cryptographic protocols** protect the exchange of money between the smart card and the accepting machine. Examples for this are **Proton**, **GeldKarte**, **Moneo** and **Quick**.

Smartcards have been advertised as suitable for these tasks, because they are engineered to be **tamper resistant**. The embedded chip of a smart card normally implements some **cryptographic algorithm**. Information about the inner workings of this algorithm can be obtained if the precise time and **electrical current** required for certain encryption or decryption operations is measured. A number of research projects have now demonstrated the feasibility of this line of attack. Counter measures have been proposed.

Another problem of smart cards may be the failure rate. The plastic card in which the chip is embedded is fairly flexible, and first time users are insufficiently careful with their card. Smart cards are often carried in wallets or pockets, which is a fairly harsh environment for a chip. However, for large banking systems, the failure management cost is more than compensated by the fraud cost reduction

CURRICULUM VITAE:

Personal Address: via Livorno n°54, Chiesanuova (BS) 25125
Telephone number: 030-3540447/ cell. 3482102835
E-mail: linja@inwind.it
Age: 18 (date of birth: 28 January 1986)
Nationality: Italian
Marital status: single

Education

2000-2003 I.P.S.I.A. Moretto
(Istituto Professionale Stato Industria e Artigianato),
Brescia
Qualification medium level: electric and electronics
technical of industry.

1996-1999 branch Franchi Secondary School, Chiesanuova (BS).

Employment

Jul-Aug 2003 Service on MOTOROLA, Via Milano (BS)
Stage experience job organized by I.P.S.I.A. Moretto,
cellular repair and maintenance Motorola, Nokia, Siemens

Interests

Computer, internet(chat).
Sport (wiet-vo-dao vietnamiti martial limbs which characteristic comprise
the body to drawn near body and the use of the crews).

Other Points

Driving licence

References

Mr. Azzani Cleto,
Electronic engineer,
Head of electronic dept,
I.P.S.I.A. Moretto,
electronic Institute,
Apollonio Street,
Brescia.