



Italian Black Hats Speech

(INFOSECURITY ITALIA 2002)

Italian Black Hats Association

<http://www.blackhats.it>

Introduzione alle Smart Cards

(Java Card)

Milano, 24/1/2002, Sala Cadamosto

Copyright

Questo insieme di trasparenze è protetto dalle leggi sul copyright e dalle disposizioni dei trattati internazionali.

Il titolo ed i copyright relative alle trasparenze (ivi inclusi, ma non limitatamente a, ogni immagine, fotografia, animazione, video e testo) sono di proprietà degli autori indicati.

Le trasparenze possono essere riprodotte ed utilizzate liberamente dagli istituti di ricerca, scolastici ed universitari afferenti al Ministero della Pubblica Istruzione per scopi istituzionali, non a fine di lucro.

Ogni altro utilizzo o riproduzione (ivi incluse, ma non limitatamente a, le riproduzioni a mezzo stampa, su supporti magnetici o su reti di calcolatori) in toto o in parte è vietata, se non esplicitamente autorizzata per iscritto, a priori, da parte dell'autore.

L'informazione contenuta in queste trasparenze è ritenuta essere accurata alla data della pubblicazione. Essa è fornita per scopi meramente didattici e non per essere utilizzata in progetti di impianti, prodotti, ecc.

L'informazione contenuta in queste trasparenze è soggetta a cambiamenti senza preavviso. Gli autori non si assumono alcuna responsabilità per il contenuto di queste trasparenze (ivi incluse, ma non limitatamente a, la correttezza, completezza, applicabilità ed aggiornamento dell'informazione).

In ogni caso non può essere dichiarata conformità all'informazione contenuta in queste trasparenze.

In ogni caso questa nota di copyright non deve mai essere rimossa e deve essere riportata anche in utilizzi parziali.

▶ INDEX

• Che cosa è una smart-card

- Struttura fisica
- Struttura logica
- Meccanismo di comunicazione
- Perché smart-card
- Standard
- Applicazioni attuali

• Attacchi alle smart card

- Invasivi
- Non invasivi

• Javacard

- Definizione
- Perché Javacard?
- SUN Javacard Development Kit

• In pratica

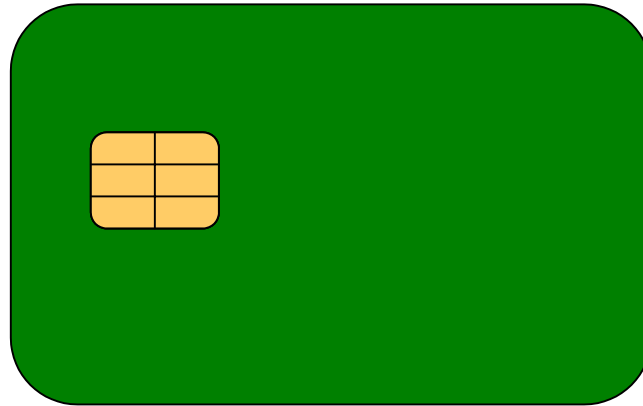
- Struttura di un'applet
- SUN Wallet

• Riferimenti

▶ SPEAKER

Francesca Fiorenza aka Kiai
Ing. Ricerca e Sviluppo

Che cosa è una Smart Card?



Un corpo di carta
Colla
Chip
Contatti

Struttura Fisica

Il microchip contiene:

Un microprocessore (generalmente 8 bits)

Memoria ROM

Memoria EEPROM

Memoria RAM

Configurazione di un microchip di gamma media:

32K ROM, 32K EEPROM, 1K RAM

32K ROM, 16K EEPROM, 2K RAM

Un co-processore crittografico

Struttura Fisica

Per funzionare la smart card deve essere inserita in un CAD

Card Acceptance Device



Letttore



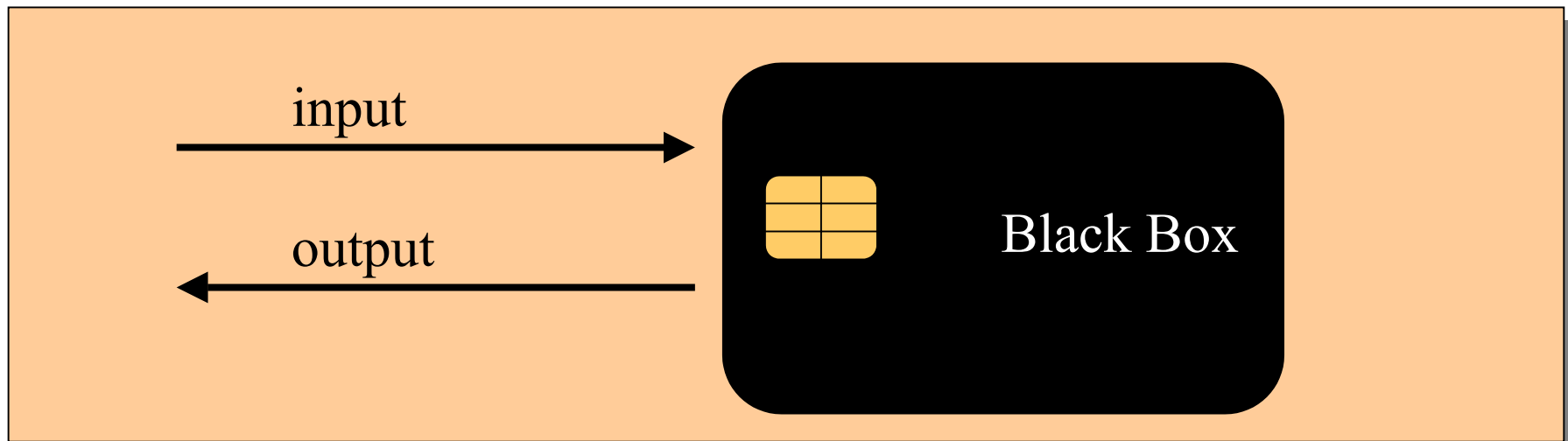
Terminale

Struttura Logica

Una smart card è una scatola nera: riceve un input, lo processa e restituisce un output

I dati non sono mai direttamente raggiungibili

Il flusso dati è bidirezionale, half duplex



Le smart card sono usate come contenitori di dati

APDU

APDU di comando

Intestazione obbligatoria

Corpo opzionale

CLA

INS

P1

P2

Lc

DATI

Le

APDU di risposta

Corpo opzionale

Traccia obbligatoria

DATI

SW1

SW2

Perché Smart Card?

Possono essere utilizzate in svariate applicazioni commerciali



- Privacy
- Integrità
- Non-repudiazione
- Autenticazione
- Verifica

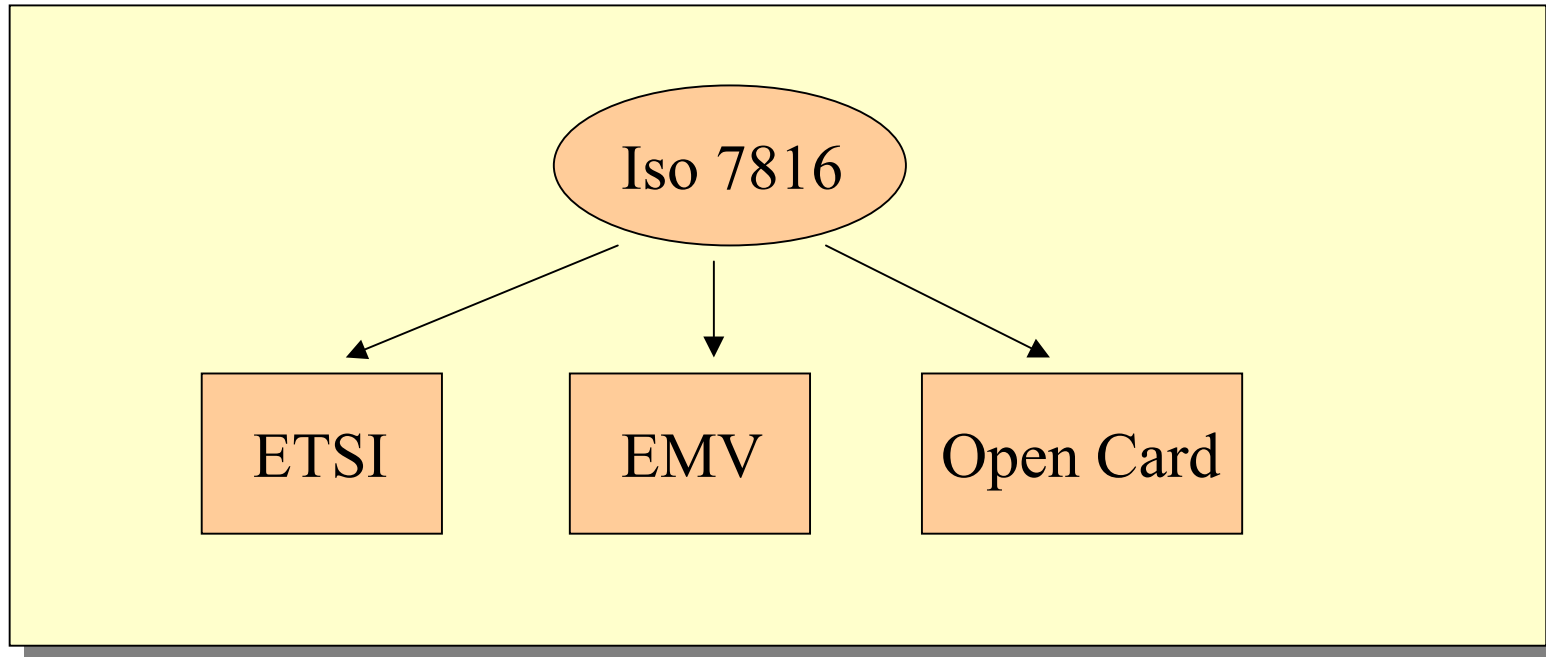
Perché sono sicure:

- Sono costituite da un solo chip
- Il processo di fabbricazione è molto sofisticato ed estremamente controllato
- Il chip è estremamente specializzato ed estremamente fragile
- Sono programmate in modo specifico
- I dati vengono protetti tramite algoritmi crittografici

Standard

Grande interesse nella standardizzazione

Garantire una lunga durata del sistema ed una interoperabilità di componenti di differenti costruttori



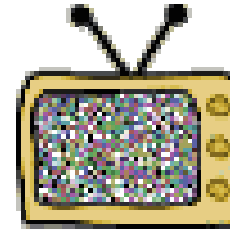
Tutte queste normative sono di pubblico dominio

Applicazioni attuali

GSM



Pay TV



Schede Telefoniche



Carte di Credito

Tessere sanitarie



Attacchi alle smart card (Invasivi)

Esigono ore in laboratori specializzati

Sono irreversibili: nel processo distruggono l'hardware

Il proprietario della carta si accorge facilmente dell'attacco e revoca le chiavi

Microprobing

Accedere alla superficie del chip

Manipolare ed interferire direttamente con i circuiti integrati

Attacchi alle smart card (non invasivi)

Molto difficili da individuare

Software

Sono gli attacchi più comuni e meno costosi.
Sfruttano la vulnerabilità dei protocolli, degli algoritmi crittografici ecc.

Eavesdropping

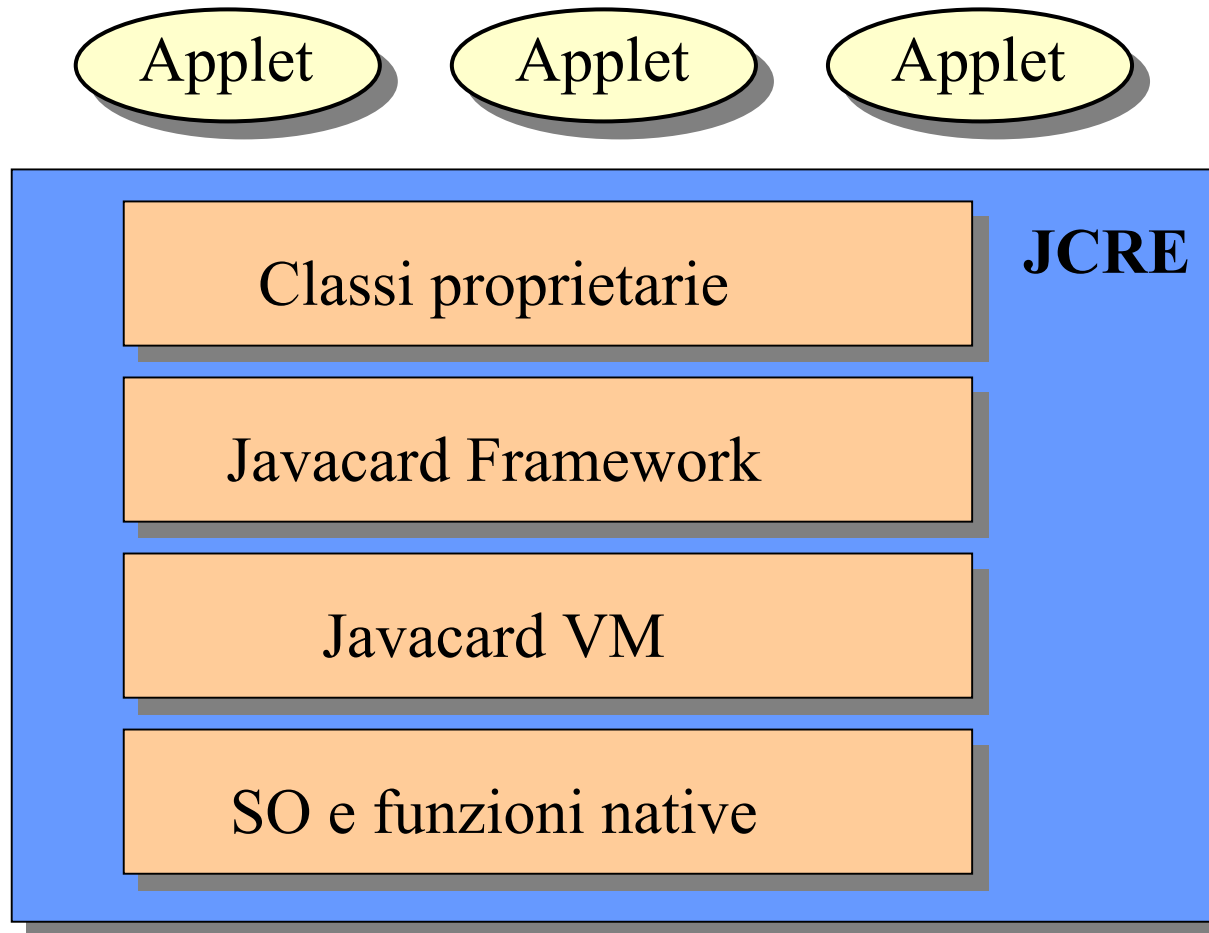
Analizzare tutte le caratteristiche delle connessioni e le radiazioni elettromagnetiche della CPU

Fault Generation

Indurre dei malfunzionamenti nel processore

Java Card

Una java card è una smart card che è capace di eseguire programmi scritti in linguaggio Java



Perché Java Card?



SICUREZZA

- Restano validi tutti i meccanismi di protezione di Java
- Eseguiti a runtime controlli di sicurezza aggiuntivi

- Indipendente dalla piattaforma
- Multiapplicativo
- Caricamento di applicazioni a posteriori
- Flessibile
- Compatibile con gli standard esistenti

SUN Java Card Development Kit

<http://java.sun.com/products/javacard/>

- ☒ Un simulatore: JCWDE
- ☒ Un convertitore
- ☒ Un verificatore off-card di bytecode
- ☒ Un installer
- ☒ Un tool per inviare comandi alla carta
- ☒ Librerie

In pratica: costruiamo un'applet

Class Applet

Superclasse di tutte le applet che risiedono sulla carta

Definisce i metodi che un'applet deve supportare per interagire con la JCRE

Select()

Chiamato dalla JCRE per informare un'applet che è stata selezionata

Install()

Chiamato dalla JCRE per creare un'istanza di un'applet

Register()

Registrare l'istanza dell'applet nella JCRE ed assegnare l'AID di default

Process()

Chiamato dalla JCRE per processare un APDU in arrivo

In pratica: SUN Wallet

Applicazione portafoglio

Addebitare/Accreditare denaro

Controllare il bilancio

Verificare il PIN

In pratica: SUN Wallet

```
public class Wallet extends javacard.framework.Applet {  
    <dichiarazione di costanti>  
    OwnerPIN pin;  
    short balance;  
  
    private Wallet (byte[] bArray, short bOffset, byte bLength) {  
        <allocazione della memoria usata dall'applet>  
        register();  
    }  
    <metodo install>  
    <metodo select>  
    <metodo process>  
}
```

In pratica: metodo *process*

| | | | | | | |
|-----|-----|----|----|----|------|----|
| CLA | INS | P1 | P2 | Lc | DATA | Le |
|-----|-----|----|----|----|------|----|

```
public void process (APDU apdu)
{
    byte[] buffer = apdu.getBuffer();
    <controlla il campo CLA dell'header>
    switch (buffer.[OFFSET_INS])
    {
        case VERIFY:           verify(apdu);           return;
        case CREDIT:           credit(apdu);           return;
        case DEBIT:             debit(apdu);           return;
        case GET_BALANCE:      getBalance(apdu);       return;
        default:
            <lancia l'eccezione comando non supportato>
    }
}
```

In pratica: Verify

| | | | | | |
|-----|-----|----|----|----|---------------------|
| CLA | INS | P1 | P2 | Lc | DATA (<i>PIN</i>) |
|-----|-----|----|----|----|---------------------|

```
private void verify (APDU apdu)
{
    byte[] buffer = apdu.getBuffer();
    byte byteRead = (byte) ( apdu.setIncomingAndReceive() );
    if(pin.check(buffer, OFFSET_DATA, byteRead) == false)
        <lancia l'eccezione verifica di PIN fallita>
}
```

In pratica: metodo Process

```
public void process (APDU apdu)
{
    byte[] buffer = apdu.getBuffer();
    <controlla il campo CLA dell'header>
    switch (buffer.[OFFSET_INS])
    {
        case VERIFY:                verify(apdu);                return;
        case CREDIT:                credit(apdu);                return;
        case DEBIT:                  debit(apdu);                  return;
        case GET_BALANCE:            getBalance(apdu);            return;
        default:
            <lancia l'eccezione comando non supportato>
    }
}
```

In pratica: Credit

| | | | | | |
|-----|-----|----|----|----|-------------------------|
| CLA | INS | P1 | P2 | Lc | DATA (<i>credito</i>) |
|-----|-----|----|----|----|-------------------------|

```
private void credit ( APDU apdu )
{
    if ( !pin.isValidated() )
        <lancia l'eccezione verifica di PIN necessaria>
    byte[] buffer = apdu.getBuffer();
    <controlla nell'header quanti byte stanno per arrivare>
    byte byteRead = (byte) ( apdu.setIncomingAndReceive() );
    <controlla che i byte arrivati siano gli stessi dell'header>
    byte creditAmount = buffer[OFFSET_DATA]
    if ( (creditAmount > MAX_TRANS_AMOUNT) || (creditAmount < 0) )
        <lancia l'eccezione importo di transazione non valido>
    if( (short) (balance + creditAmount) > MAX_BALANCE)
        <lancia l'eccezione importo superiore al bilancio massimo>
    balance = (short) (balance + creditAmount);
}
```

In pratica: metodo Process

```
public void process (APDU apdu)
{
    byte[] buffer = apdu.getBuffer();
    <controlla il campo CLA dell'header>
    switch (buffer.[OFFSET_INS])
    {
        case VERIFY:           verify(apdu);           return;
        case CREDIT:          credit(apdu);           return;
        case DEBIT:             debit(apdu);           return;
        case GET_BALANCE:     getBalance(apdu);       return;
        default:
            <lancia l'eccezione comando non supportato>
    }
}
```

In pratica: Debit

| | | | | | |
|-----|-----|----|----|----|------------------------|
| CLA | INS | P1 | P2 | Lc | DATA (<i>debito</i>) |
|-----|-----|----|----|----|------------------------|

```
private void debit ( APDU apdu )
{
    if ( !pin.isValidated( ) )
        <lancia l'eccezione verifica di PIN necessaria>
    byte[] buffer = apdu.getBuffer();
    <controlla nell'header quanti byte stanno per arrivare>
    byte byteRead = (byte) ( apdu.setIncomingAndReceive() );
    <controlla che i byte arrivati siano gli stessi dell'header>
    byte debitAmount = buffer[OFFSET_DATA]
    if ( ( debitAmount > MAX_TRANS_AMOUNT ) || ( debitAmount < 0 ) )
        <lancia l'eccezione importo di transazione non valido>
    if( (short) (balance - debitAmount) < (short) 0 )
        <lancia l'eccezione bilancio negativo>
    balance = (short) ( balance - debitAmount );
}
```

In pratica: metodo Process

```
public void process (APDU apdu)
{
    byte[] buffer = apdu.getBuffer();
    <controlla il campo CLA dell'header>
    switch (buffer.[OFFSET_INS])
    {
        case VERIFY:           verify(apdu);           return;
        case CREDIT:          credit(apdu);           return;
        case DEBIT:           debit(apdu);           return;
        case GET_BALANCE:      getBalance(apdu);      return;
        default:
            <lancia l'eccezione comando non supportato>
    }
}
```

In pratica: Get Balance

| | | | | |
|-----|-----|----|----|----------|
| CLA | INS | P1 | P2 | Le (= 2) |
|-----|-----|----|----|----------|

```
private void getBalance( APDU apdu )
{
    byte[] buffer = apdu.getBuffer();
    short le = apdu.setOutgoing();
    if ( le < 2 )
        <lancia l'eccezione lunghezza sbagliata>
    apdu.setOutgoingLength( (byte) 2 );
    buffer[0] = (byte) (balance >> 8);
    buffer[1] = (byte) (balance & 0xFF);
    apdu.sendBytes( (short) 0, (short) 2 );
}
```

Riferimenti

- **www.oberthurcs.com**

- <http://java.sun.com/products/javacard/>

- <http://java.sun.com/products/commerce/>

- <http://www.smartcardcentral.com/>

- <http://www.opencard.org/>

- <http://www.iso.ch>

- <http://www.etsi.org>

- <http://csrc.ncsl.nist.gov/>

- <http://www.faqs.org/faqs/technology/smartcards/faq/index.html>

- alt.technology.smartcards

Italian Black Hats Association

Introduzione alle Smart Cards **(Java Card)**

24 gennaio 2002, Infosecurity Italia

Relatore:

Francesca Fiorenza aka Kiai@blackhats.it

BlackHats.it

General Infos:
info@blackhats.it