

presented by



Supporting Smart Cards in UEFI

UEFI PlugFest– March 18-22, 2013
Presented by Jean Lusetti (Gemalto)

Agenda



- Who is Gemalto ?
- What is a smart card?
- Why smart cards in UEFI?
- Adding smart card support
 - Understand our needs
 - Raw draft proposal
 - Comparison
- What's next
- Questions

Gemalto

Securing Digital Interactions – Everywhere



Vision

In an increasingly connected society,
Gemalto will be the **leader**
in making personal digital interactions
secure and easy – wherever you are

-
- ✦ €2.2bn revenues 2012
 - ✦ >10,000 employees
 - ✦ >100 nationalities
 - ✦ >145 offices & facilities

FINANCIAL SERVICES & RETAIL

GOVERNMENT

TRANSPORT

ENTERPRISE

TELECOMMUNICATIONS

MACHINE-TO-MACHINE

Financial services & retail

Trusted and easy ways to bank & pay



Banks need solutions that

- ✦ Reduce fraud at ATM, in store & online
- ✦ Help replace cash
- ✦ Create trust in their web services
- ✦ Make payment more convenient & more mobile



Here's our answer

- Software & services for card personalization & issuance
- Payment & loyalty cards
- Security for online banking
- Contactless payment and mobile solutions



Government

Identifying citizens & powering eGovernment



Governments need solutions that

- ✦ Help them migrate their traditional identity & travel documents to electronic formats
- ✦ Reduce costs while deploying 24/7 eGovernment services
- ✦ Remove paperwork & ensure citizens can access their rights & benefits e.g. healthcare



Here's our answer

- Solutions for citizen data enrolment, ID issuance, border control & eGovernment
- Secure documents for travel, ID, eHealth, drivers licenses & registration certificates (birth etc.)
- Outsourced personalization & issuance services

Transport

Helping commuters beat the lines



Transport operators need to

- ✦ Replace paper tickets & cash while looking to multi-application and mobile ticketing solutions
- ✦ Deploy contactless travel passes and mobile ticketing as part of our trusted service management offer, such as the first mobile NFC service in Nice, France
- ✦ Help their customers commute in 30 cities including Rio, London & Paris



Here's our answer

- Our contactless cards are used to access mass transit systems in cities of over one million inhabitants all over the world
- To date we've rolled out more than 140 million contactless travel cards

Enterprise security solutions

Simplifying security for networks & businesses



Organizations need solutions that

- ✘ Safeguard IT network access
- ✘ Allow only authorized people to enter their buildings
- ✘ Secure access to data from anywhere including cloud-based apps



Here's our answer

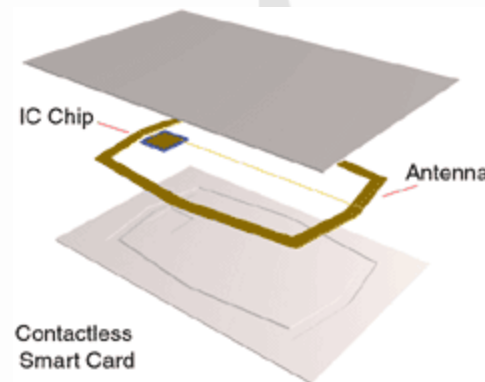
- Full suite of ID products (cards, tokens etc.) for strong authentication
- Devices for secure storage & data encryption (even email)
- Authentication as server software or as a service



What is a smart card?



- Sort of mini-computer
 - CPU (8, 16 or 32bit)
 - RAM (~4 KB)
 - ROM (~256 KB)
 - EEPROM (~50 KB)
 - Optional Crypto-Processor
 - Operating System
 - Run applets
 - Java Card
 - .NET
 - Native



Some use-cases



- A SIM in your mobile phone ([ETSI](#))
 - The smart card authenticates you to the your GSM provider
- In your credit card ([EMV](#))
 - To secure transactions
 - To certify you made it
 - To secure transactions over the Internet
- In your company
 - For strong authentication using PKI
 - To sign/cipher emails
 - For physical access (contactless)
- When travelling/today's life
 - A contactless smart card in your passport
 - In your ID card, driver license
 - For health-care



Converged Badge



TELECOMMUNICATIONS

FINANCIAL SERVICES & RETAIL

ENTERPRISE

TRANSPORT

GOVERNMENT

Some figures



- How many smart cards shipped in **2012**?

➤ >7,000,000,000 Yes, billions!



- Each year, one smart card shipped for every single human being
- Business segmentation
 - 75% : Telecom
 - 16% : Payment



Why smart cards in UEFI?



- Some smart card applications in UEFI today
 - All proprietary
 - No interoperability
 - May use dedicated features for a given context
 - Time to have a standardized framework !
- Main use-case : Pre-boot Authentication
 - Add 2/3 factor authentication (FA) to UEFI
 - Something you have (smart card)
 - Something you know (PIN code)
 - Something you are : Bio (e.g. fingerprint)
 - Smart card can feature Match on card
 - First step for Single Sign On with OS
 - As a natural extend to PreBoot/OS seamless experience

Why smart cards in UEFI?



- Protect BIOS configuration
 - Sensitive options access granted by smart card
- Disk encryption
 - Using keys protected by smart card
- Test smart card readers
 - No need for an OS dependant test application

Existing standards

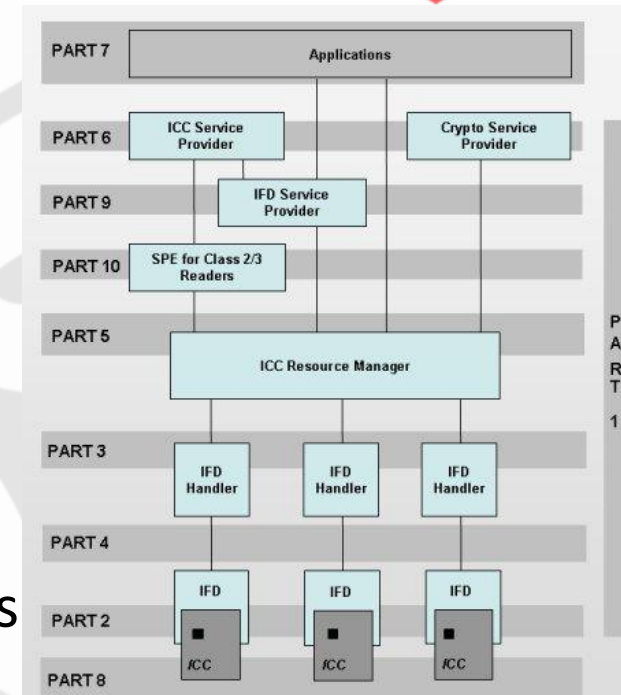


- ISO7816-x
- [EMV](#) (Banking)
 - Eurocard, Mastercard, Visa consortium
- [USB CCID](#)
 - Circuit(s) Cards Interface Devices
- [PC/SC Workgroup](#)
 - Smart card support in Operating Systems

Adding smart card support



- PC/SC specifications
 - Divided into 10 Parts
- Resource Manager
 - Main component
 - Exposes API (Part 5)
 - Responsible for resource sharing
 - Populates readers/cards to applications
 - Implemented as a service or daemon
- IFD Handler
 - Device driver for the smart card reader, usually CCID



Adding smart card support



- Main objectives
 - Be as close as possible to PC/SC API Part 5
 - Make it as simple as possible
- So let's make some choices!
 - Resource Manager main jobs
 - Expose resources to applications
 - Let's use UEFI protocol discovery instead
 - Resource sharing/service
 - No need for that in UEFI
 - Focus on Part 5
 - Let IFD handler expose Part 5, remove Part 3
 - Keep only main functions

Let's clean up Part 5!



RESOURCE MANAGER	
EstablishContext	X
ReleaseContext	X

RESOURCE DB	
IntroduceReader	X
ForgetReader	X
IntroduceReaderGroup	X
ForgetReaderGroup	X
AddReaderToGroup	X
RemoveReaderFromGroup	X
IntroduceCardType	X
ForgetCardType	X

SCARD TRACK	
LocateCards	X
GetStatusChange	X
Cancel	X

RESOURCE QUERY	
ListReaderGroups	X
ListReaders	X
ListCardTypes	X
GetProviderId	X
ListInterfaces	X

SCARD COMM	
Connect	X
Reconnect	X
Disconnect	X
Status	X
BeginTransaction	X
EndTransaction	X
Cancel	X
Transmit	X
Control	X
GetReaderCapabilities	X
SetReaderCapabilities	X

EFI_BOOT_SERVICES
LocateHandleBuffer
OpenProtocol

EFI_SMART_CARD_READER_PROTOCOL
Connect
Disconnect
Status
Transmit
Control
GetReaderCapabilities

29 → 6+2 functions

UEFI vs original API



```
EFI_STATUS SCardStatus(  
    EFI_SMART_CARD_READER_PROTOCOL *This,  
    OUT CHAR8 *ReaderName,  
    IN OUT UINTN *ReaderNameLength,  
    OUT UINT32 *State,  
    OUT UINT32 *CardProtocol,  
    OUT UINT8 *Atr,  
    IN OUT UINTN *AtrLength  
)
```

CardExclusive,
Reader

NoReset,
ColdReset,
WarmReset

```
EFI_STATUS SCardConnect(  
    EFI_SMART_CARD_READER_PROTOCOL *This,  
  
    IN UINT32 Flags,  
    IN UINT32 PreferredProtocol,  
  
    OUT UINT32 *ActiveProtocol  
)
```

```
LONG SCardStatus(  
    IN SCARDHANDLE hCard,  
    OUT LPTSTR szReaderName,  
    IN OUT LPDWORD pcchReaderLen,  
    OUT LPDWORD pdwState,  
    OUT LPDWORD pdwProtocol,  
    OUT LPBYTE pbAtr,  
    IN OUT LPDWORD pcbAtrLen  
)
```

CardShare,
CardExclusive,
Reader

```
LONG SCardConnect(  
    IN SCARDCONTEXT hContext,  
    IN LPCTSTR szReader,  
    IN DWORD dwShareMode,  
    IN DWORD dwPreferredProtocols,  
    OUT LPSCARDHANDLE phCard,  
    OUT LPDWORD pdwActiveProtocol  
)
```

API use



Typical PC/SC sequence call

```
SCardEstablishContext(..., &hContext);
```

```
SCardListReaders(hContext, ..., ReaderNames);  
ReaderName=ReaderNames[n];
```

```
SCardConnect(ReaderName, ..., &hCard);
```

```
SCardTransmit(hCard, CAPDU,..., RAPDU);
```

```
SCardDisconnect(hCard,...);
```

```
SCardReleaseContext(hContext);
```

UEFI equivalence

```
gBS->LocateHandleBuffer(..., hReaders);  
gBS->OpenProtocol(hReaders[n], &ReaderProtocol, ...);  
ReaderProtocol->SCardStatus(..., ReaderName, ...);  
gBS->FreePool (hReaders);
```

```
ReaderProtocol->SCardConnect(...);
```

```
ReaderProtocol->SCardTransmit(CAPDU,..., RAPDU);
```

```
ReaderProtocol->SCardDisconnect(...);
```

```
gBS->CloseProtocol(ReaderHandle,...);
```




UEFI restrictions for PC/SC

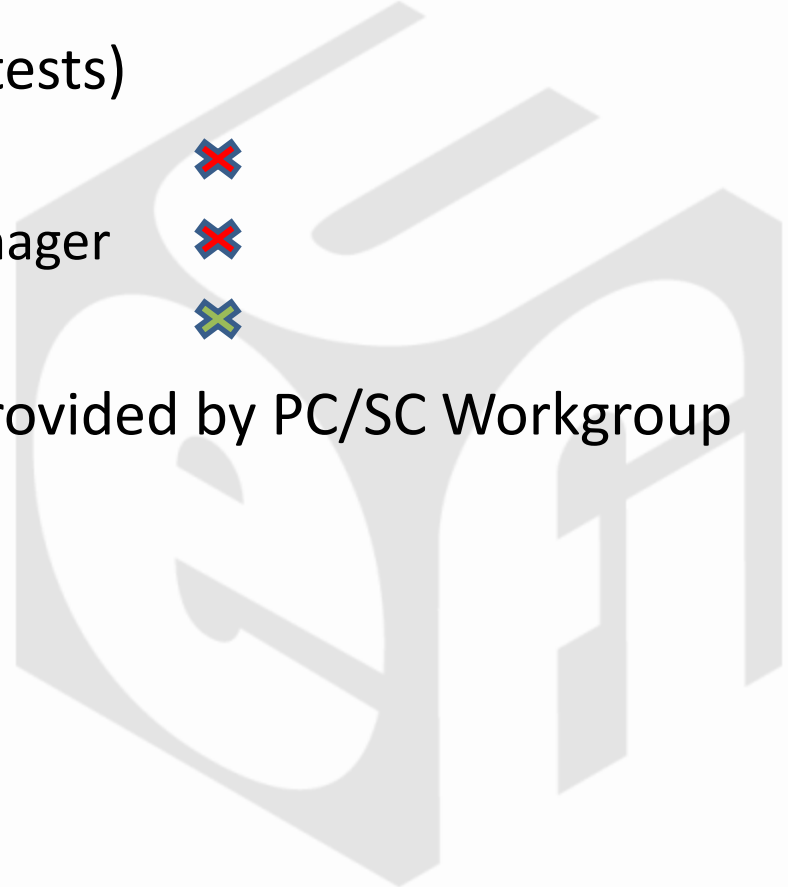


- No Resource Manager
 - No card connection sharing between applications
 - Should not be an issue as UEFI is not multi-tasking
- Power ON/OFF cycles
 - Used to be controlled by Resource Manager
 - Makes sense in a multi-app context
 - 2 possibilities now:
 - UEFI applications now have full control
 - Driver can control these cycles
- Reader selection is a bit different

Testing



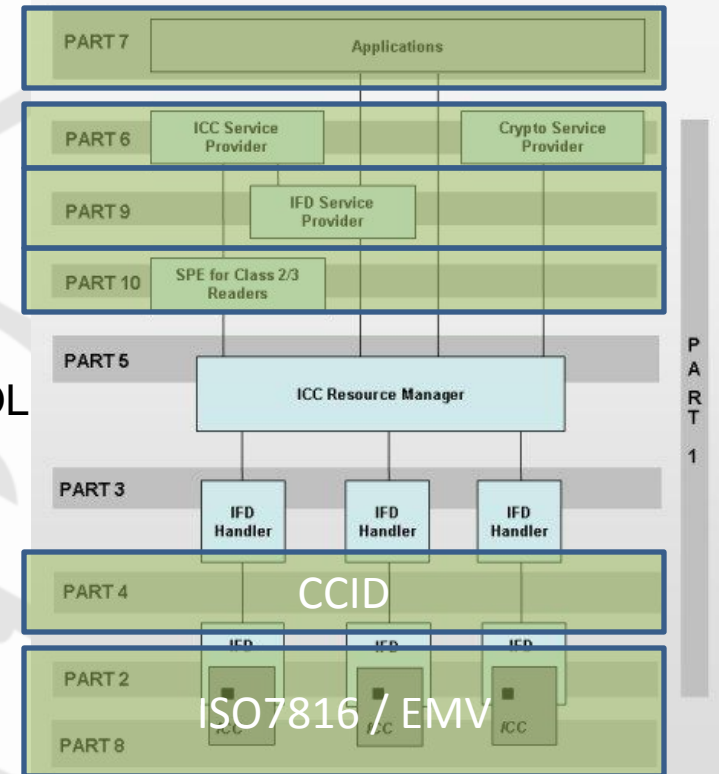
- Get inspired by Microsoft HCK for smart card readers
- Limit it to Part D (smart card tests)
 - Power Management 
 - Interface with Resource Manager 
 - Card insertion/removal 
- Leverage on smart card set provided by PC/SC Workgroup



What's next



- Make the official proposal for EFI_SMART_CARD_READER_PROTOCOL
 - Discuss it in USWG
- OK for parts 3 and 5, what about others?
 - ✓ Part 6
 - Smart card file access and authentication APIs
 - Many standards here
 - PKCS#11
 - CAPI (Microsoft)
 - EFI_USER_CREDENTIAL2_PROTOCOL
 - Can live without it at least at the beginning
 - ✓ Part 10 is implemented in Part 3 (PC/SC history)
 - ✓ Other Parts are relying on other specifications (CCID, ISO7816,...), are poorly used or just recommendations



Thanks for attending the
UEFI Spring PlugFest 2013



For more information on
the Unified EFI Forum and
UEFI Specifications, visit
<http://www.uefi.org>

www.justaskgemalto.com

www.twitter.com/gemalto



www.gemalto.com

<http://blog.gemalto.com>

presented by

