Smart Cards and Crypto Devices

Smart cards? Well, a certain animal cunning perhaps.

Smart Cards

Invented in the early 1970s

- Technology became viable in the early 1980s

Cool idea for the 1970s, but a *terrible* handicap on the technology

- You simply cannot make a credit-card form factor device robust, capable, or secure

Smart Cards (ctd)

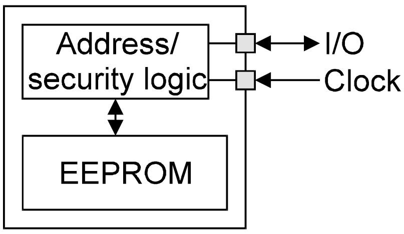
Major use is in prepaid telephone cards (hundreds of millions)

- Use a one-way (down) counter to store the card balance
- Even then it took years to get them right

Other uses

- Student ID/library cards
- Patient data
- Micropayments (bus fares, photocopying, snack food)

Memory Cards



Usually based on I^2C (serial memory) bus

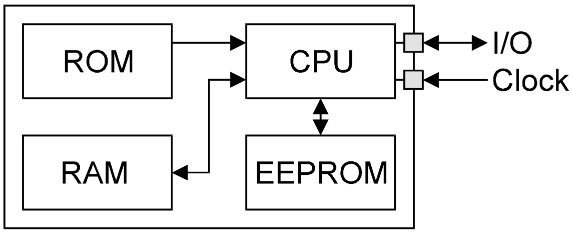
Typical capacity: 256 bytes

EEPROM capabilities

- Nonvolatile storage
- 10,000 write/erase cycles
- 10ms to write a cell or group of cells

Cost: $5

Microprocessor Cards



ROM/RAM contains card operating system and working storage

EEPROM used for data storage

Microprocessor Cards (ctd)

Typical specifications

- 8-bit CPU
  - Advertised as 16-bit by combining 8-bit register pairs
- 16K-32K ROM
- 256-512 bytes RAM
- 4K-16K EEPROM
  - Advertised in bits to make it sound bigger

Size ratio of memory cells:

RAM = 4× EEPROM size
= 16× ROM size

- Everything has to be fabbed on the same die

Cost: $5-50 (with crypto accelerator)

Smart Card Technology

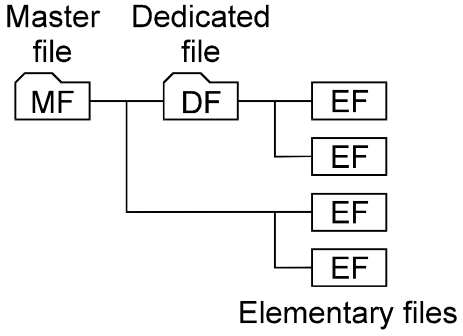
Based on the ISO 7816 standard, which defines

- Card size, contact layout, electrical characteristics
- I/O protocols
  - Byte-based
  - Block-based
- File structures

Terminology alert: Vendor literature often misuses standard terms for advertising purposes

- “16-bit” = 2 × 8-bit register pairs
- “Digital signature” = simple checksum or MAC
- “Certificate” = data + “digital signature”

File Structures



Files addressed by 16-bit file ID (FID)

- FID is often broken up into DF:EF parts (MF is always 0x3F00)

Files are generally fixed-length and fixed-format

File Types

Transparent

- Binary blob

Linear fixed

- $n \times$ fixed-length records

Linear variable

- n records of fixed (but different) lengths

Cyclic

- Linear fixed, oldest record gets overwritten

Execute

- Special case of transparent file

File Attributes

EEPROM has special requirements (slow write, limited number of write cycles) that are supported by card attributes

- WORM, only written once
- Multiple write, uses redundant cells to recover when some cells die
- Error detection/correction capabilities for high-value data
- Error recovery, ensures atomic file writes
  - Power can be removed at any point
  - Requires complex buffering and state handling

Card Commands

Typical commands are

- CREATE/SELECT/DELETE FILE
- READ/WRITE/UPDATE BINARY
  - Write can only change bits from 1 to 0 because of EEPROM technology limits
  - Update is a genuine write
- ERASE BINARY
- READ/WRITE/UPDATE RECORD
- APPEND RECORD
- INCREASE/DECREASE
  - Changes cyclic file position

Card Commands (ctd)

Access control

- Based on PIN or chip holder verification (CHV)
- VERIFY CHV
- CHANGE CHV
- UNBLOCK CHV
- ENABLE/DISABLE CHV

Authentication

- Simple challenge/response authentication protocol
- INTERNAL AUTHENTICATE
  - Authenticate card to terminal
- EXTERNAL AUTHENTICATE
  - Authenticate terminal to card

Card Commands (ctd)

Encryption: Various functions, typically

- ENCRYPT/DECRYPT
- SIGN DATA/VERIFY SIGNATURE

Electronic purse instructions

- INITIALISE/CREDIT/DEBIT

Application-specific instructions

- RUN GSM ALGORITHM
- prEN 1546 commands INITIALISE IEP, CREDIT IEP, DEBIT IEP, CONVERT IEP CURRENCY, and UPDATE IEP PARAMETER

Working with Cards

ISO 7816 provides only a standardised command set, implementation details are left to vendors

- Everyone does it differently
- Cards are made gratuitously incompatible to force customer lock-in

Standardised API's were quite slow to appear

- PKCS #11, general API for any crypto device
- PC/SC, Windows HAL for smart cards
- JavaCard, Java-like language for restricted environments

Working with Cards (ctd)

The Smart Card Problem

- No cards
- No readers
- No software

Installation of readers and cards is too problematic

- Keyboard and mouse (or all of Windows) may stop working
- Installing more than one reader, or reinstalling/updating drivers, is a recipe for disaster
  - Drivers need to be installed in exactly the right order
  - PC operations may be affected (e.g. other peripherals stop working, system functions are disabled)
  - Drivers/readers may cease to function entirely

Working with Cards (ctd)

Even finding basic DES encryption that works is tricky

- Schlumberger Cryptoflex: Doesn’t make DES user-accessible
- Schlumberger Multiflex: Returns only 6 of 8 encrypted bytes
- IBM MFC: Encrypts a random number
- Maosco MULTOS: Uses a fixed, known key “for security reasons”
- General Information Systems OSCAR: XORs the DES key with a random number “for security reasons”
- Gemplus GPK: Restricts keys to 40 bits

PKCS #11

Object-oriented interface to any type of crypto token

- Smart card
- Crypto hardware accelerator
- Fortezza card
- USB-based token
- Handheld PC (e.g. PalmPilot)
- Software implementation

Programming interface is completely independent of the underlying token type

PKCS #11 (ctd)

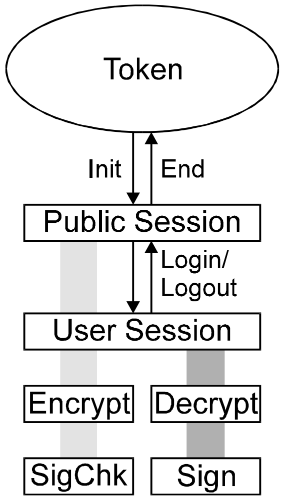
Token provides various services to the caller

- Store public/private keys, certificates, secret keys, authentication values, generic data
- Encrypt/decrypt
- Sign/signature check
- Wrap/unwrap key
- Generate key, generate random data
- Find object in token

PKCS #11 (ctd)

Services can be restricted until the user has logged on using a PIN/password

- This is an abstraction only, tokens may implement this in various ways
  - Public session has no access, only user sessions are recognised
  - User session has full access, no concept of public vs. user



PKCS #11 Token Objects

Token objects are structured in a hierarchical manner

- Object
  - Key
    - Public Key
      - RSA Public Key
      - DSA Public Key
      - DH Public Key
    - Private Key
      - RSA Private Key
      - DSA Private Key
      - DH Private Key
    - Secret Key
      - DES Key
      - 3DES Key
      - RC2/RC4/RC5 Key
  - Certificate
    - X.509 Certificate
  - Data

PKCS #11 Token Objects (ctd)

Each object has a collection of attributes that follow the token object hierarchy

Example: RSA private key

- Object attributes

CKA_CLASS = CKO_PRIVATE_KEY	
CKA_TOKEN = TRUE	(persistent object)
CKA_PRIVATE = TRUE	(needs login to use)
CKA_MODIFIABLE = FALSE	(can’t be altered)
CKA_LABEL = “My private key”	(object ID for humans)

- Key attributes

CKA_KEY_TYPE = CKK_RSA	
CKA_ID = 2A170D462582F309	(object ID for computers)
CKA_LOCAL = TRUE	(key generated on token)

PKCS #11 Token Objects (ctd)

- Private Key attributes

CKA_SENSITIVE = TRUE	(attributes can’t be revealed outside the token)
CKA_EXTRACTABLE = FALSE	(can’t be exported from the token)
CKA_DECRYPT = TRUE	(can be used to decrypt data)
CKA_SIGN = TRUE	(can be used to sign data)
CKA_UNWRAP = TRUE	(can be used to unwrap encryption keys)

- RSA Private Key attributes

  CKA_MODULUS = …
  CKA_PUBLIC_EXPONENT = …
  CKA_PRIVATE_EXPONENT = …
  CKA_PRIME_1 = …
  CKA_PRIME_2 = …
  CKA_EXPONENT_1 = …
  CKA_EXPONENT_2 = …

PC/SC

Interoperability Specification for ICC’s and Personal Computer Systems

- Microsoft’s attempt to kill PKCS #11 (c.f. PCT vs. SSL)
- Goes a long way towards solving the Smart Card Problem

PC/SC spec defines

- Physical and electrical characteristics as for ISO 7816
- Interface device (IFD) handler
  - Common software interface for card readers
  - Sets out minimal IFD requirements (command handling, card insertion check)
- Integrated circuit card (ICC) resource manager
  - Controls all IFD’s attached to the system

PC/SC (ctd)

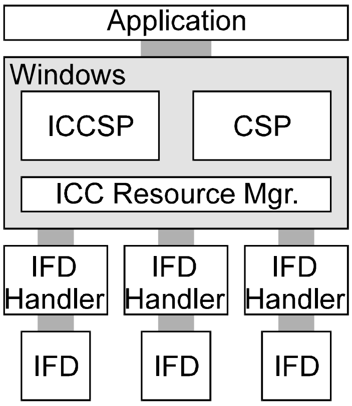
PC/SC spec (ctd)

- ICC service provider (ICCSP)
  - Maintains context of a card session
- Crypto service provider (CSP)
  - Optional manager for crypto functionality
  - Separated out for export control purposes

Getting it all to work properly was a remarkable achievement

- Only Microsoft’s muscle could finally get the card vendors to play ball

PC/SC (ctd)



Provided as an integral part of newer Windows releases

- ~~Simon~~ Bill says smartcards

JavaCard

Standard smart card with an interpreter for a Java-like language in ROM

- Card runs Java with most features (multiple data types, memory management, most class libraries, and all security (via the bytecode verifier)) stripped out
  - Can run up to 200 times slower than card native code
  - Language is pseudo-Java, not real Java
  - No security, a simple out-of-bounds memory access can destroy the code

Provides the ability to mention both “Java” and “smart cards” in the same sales literature

JavaCard (ctd)

Card contains multiple applets

- External client sends `select` command to card
- Card selects applet and invokes its `select` method
- Further commands sent by the client are forwarded to the applet's `process` method
- Applet is shut down via `deselect` method when a new select command is received

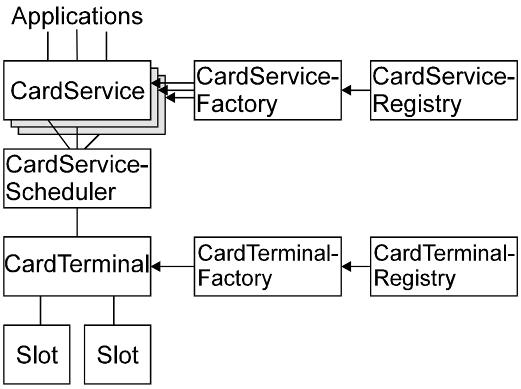
Applet can access packages and services from other applets

- How to do this securely is still under debate

OCF

Open Card Framework, object-oriented framework for smart card developers

- Class contains a blueprint for an object
- Object is an instance of a class



OCF (ctd)

class SmartCard

CardID

– Information identifying the card

CardServiceFactory

CardService: PurseCardService
CardService: FileSystemCardService
CardService: …

CardServiceRegistry

– Looks up requested CardService in CardServiceFactory

– Instantiates a new CardService object for the caller

CardServiceScheduler

– Communicates with the card terminal

– Coordinates access to card services

OCF (ctd)

class CardFile

Attributes

– TRANSPARENT, LINEAR FIXED, …

CardFilePath, CardFileInputStream, CardFileOutputStream

class Terminal

Slot

– Information on reader slot + optional display, keyboard

CardTerminalFactory

CardTerminal

CardTerminalRegistry

– As CardServiceRegistry

PKCS #11 vs. OCF vs. PC/SC

	Language	OS	Abstraction Level
PKCS #11	Any	Any	High
PC/SC	Any	Windows	Low
OCF	Java	JVM	Low

PKCS #11: Powerful but complex to implement

PC/SC: Any platform you want as long as it's Windows (limited availability under Linux)

OCF: By Java programmers for Java programmers

Multiapplication Cards

Smart cards aren't taking off

- Multiapplication smart cards may help
- "There are no applications for them now, but if you go with a multiapplication card you can charge others to put their apps in your card"

Whose name goes on the front?

Smart card internal security ranges from nonexistent (early 1950s computers) to minimal (late 1950s computers)

- Would VISA share their card with Amex?
- Would VISA share their card with Honest Bob's Used Cars?
- Which app are you communicating with?
- Who takes responsibility if something goes wrong?

Multiapplication Cards (ctd)

Problems with application deactivation

- Combined payment application and ID card → cancelling payment card revokes your electronic passport
- Combined payment application and military ID → payment system compromise can threaten military security

Differences in target market

- Credit-worthy bank customers get a smart card for their account
- Non-credit-worthy bank customers get a smart card for prepayment electricity meters
- Customer sets are mutually exclusive

Multiapplication Cards (ctd)

Different usage models

- Credit cards are handed to waiters who disappear with them
- Most people will be reluctant to hand their electronic passport/bank card to a random waiter
- Low-value payment cards (phone cards, copier cards) are commonly shared with acquaintances/family
- Problematic when the card can also perform high-value transactions

Card failure is a problem

- About 1% of cards fail each year
- Lose all of your data/money in one go

Smart Card Limitations

Typical cards have the following limits

- 9600bps only (~1K/sec communication rate)
  - Many USB devices are just serial I/O cards with USB glue
  - A single public-key operation can take seconds just to communicate the data
  - Many card CPUs are also used for I/O, card can't do anything else while communicating
- 3.5 – 5 MHz clock (slow 8-bit CPU)
- No on-board battery (power analysis attacks)
- Limited chip size ($5mm^2$) and thickness due to packaging constraints

Other crypto token form factors (PCMCIA card, iButton) avoid these problems

Dallas iButton

Avoids most smart card problems by changing the packaging

Device is contained in 16×5mm microcan

- Stainless steel case is much stronger than a smart card
- Case contains a built-in battery and clock
- I/O doesn't tie up a serial port
  - $10 iButton interface is cheaper than $50 card reader

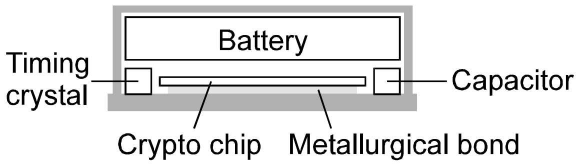
Capabilities range from simple serial-number ID, real-time clock, and data storage to crypto iButton

- 8051 processor, 32K ROM, 6K NVRAM
- 1024-bit crypto accelerator
- Real-time clock

iButtonSecurity

iButton package allows for much better security measures than smart cards

- Various triggers erase memory if tampering is detected



- Active face of chip is metallurgically bonded to the base of the can
- Energy reservoir capacitor is used to zeroise memory
- Timing crystal drives the on-board clock

iButton Security (ctd)

Zeroisation can be triggered by

- Opening the case
- Disconnecting the battery
- Temperatures below -20º C or above 70º C
- Excessive voltage levels
- Attempts to penetrate the case to get to the chip
  - Chip contains screen to prevent microprobing

iButton Programming

The device recognises two roles

- Crypto officer initialises the device
  - Create transaction group(s)
  - Set up information (keys, monetary value, etc)
  - Set initial user PIN
  - Lock transaction group(s)
- User utilises it after initialisation by the crypto officer

Device contains one default group (Dallas Primary Feature Set) initialised at manufacture

- Allows crypto officer to initialise the device
- Allows the user to verify that the crypto officer hasn’t altered certain initial options

iButton Programming (ctd)

Dallas Primary contains a default private key generated by the device at manufacture

- Corresponding public key is certified by the manufacturer
- Guarantees to a third party that a given initial key belongs to a given iButton
- Users can generate further keys as required

Provides a clear chain of custody for the device

- Dallas $\rightarrow$ crypto officer $\rightarrow$ user
- Follows the baby-duck security model used by a number of other high-security devices

iButton Special Features

Device provides enhanced signature capabilities using on-board resources

- Signing time
- Transaction counter (incremented for each signature, used to detect trojan signing software)
- Device serial number

Signing process

- User hashes data with MD5, SHA-1, RIPEMD-160, …
- iButton hashes the user-supplied hash with the device serial number, transaction counter, and timestamp
- iButton signs the hash using its private key
- User retrieves the serial number, transaction counter, timestamp, and signature from the iButton

iButton → Java iButton

It'll be kewler if we Java-ise it

- Rip out all the security features, lobotomise the security model
- Turn it into a Javacard in a can

Short-term effect

- Positive publicity because it contains the word "Java"

Long-term effect

"Look, a Java processor in a can"

"Yes, but what's it good for?"

"No, you don't understand. It's a Java processor! In a can!"

Java iButton was discontinued in late 2005

Contactless Cards

Several levels of contactless cards

- Contact, ISO 7816
- Close-coupled, 0-2mm, ISO 10536
  - Abandoned in favour of proximity cards
- Proximity, 0-10cm, ISO 14443
  - Typical use: MIFARE, transport applications
- Vicinity, ~1m, ISO 15693
  - Typical use: RFID

Terminology and specs mirror ISO 7816

- Card = Proximity Integrated Circuit Card, PICC
- Reader = Proximity Coupling Device, PCD

Contactless Cards (ctd)

Contactless card issues

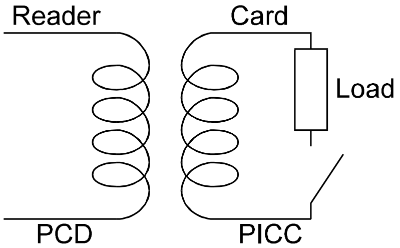
- Power and communications link is unstable
- Background noise problems
- Low power levels
  - Boosting power increases RFI caused by carrier sidebands
  - Maximum range determined by level at which RFI still complies with emission laws
- Transaction must be rapid (100-200ms)
  - Move as many people through as few turnstiles as possible

Contactless Card Communications

Power and data transmitted at 13.56 MHz

- Card coil requires only a few turns
- Coil can be a printed circuit or a standard wire coil
- Card and reader communicate at 106 Kbps (13.56MHz/128)

Card communicates with the reader using load modulation

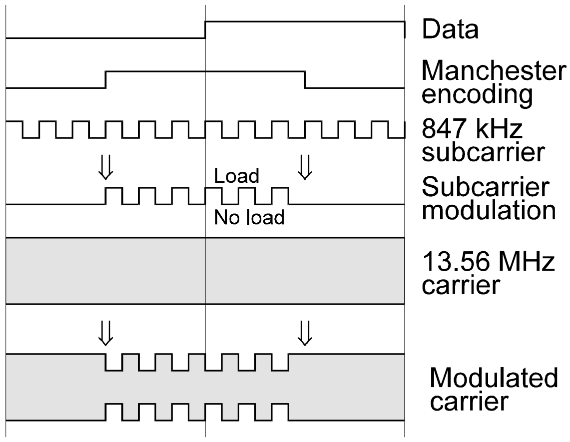


- Switches a load resistor in and out of circuit
- Reader detects the changes in load

Contactless Card Communications (ctd)

Load modulation types

- Type A, simple and efficient



- Type B, complex and inefficient — included for political reasons

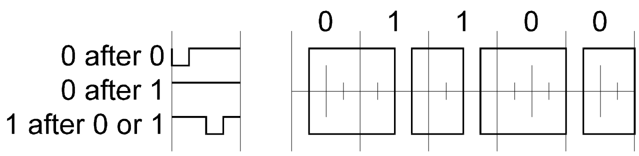
Contactless Card Communications (ctd)

Reader communicates with card using ASK

- 100% amplitude shift keying = turn carrier on and off
- CMOS circuits in the card consume no power when they're not switched

Encoding uses a modified Miller code

- Carrier pauses at different positions
- To decode, the card measures the distance between pauses

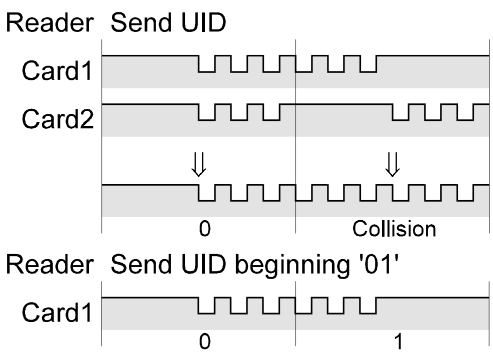


- Detecting errors like dropped bits is very simple

Initialisation and Anticollision Handling

Card initialisation

- Reader actives all cards using a wakeup frame
  - Wakeup frame has a special format to distinguish it from normal frames
- Extraneous cards are weeded out using collision detection and depth-first search
- RFID blockers possible
  - Respond to every query from the reader



Vicinity Cards

Extend proximity card ideas

- PCD → VCD (Vicinity card device)
- PICC → VICC (Vicinity integrated circuit card)

Vicinity card requirements

- Low-cost, high volume, long range, simple cards

More commonly use type B modulation

- Less RFI allows operation over longer ranges
- Use PPM (pulse position modulation) for VCD → VICC, FSK for VICC → VCD
  - Communication rate 6.6 Kbps
  - Variations on modulation, coding, and baud rate for different applications (speed vs. distance vs. noise immunity vs. emission levels)

Attacks on Smart Cards

Use a doctored terminal/card reader

- Reuse and/or replay authentication to the card
- Display \$*x* transaction but debit \$*y*
- Debit the account multiple times

Protocol attacks

- Card security protocols are often simple and not terribly secure

Fool CPU into reading from external instead of internal ROM

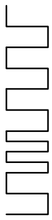
Attacks on Smart Cards (ctd)

Manipulating supply voltages can affect security mechanisms

- Picbuster

Clock/power glitches can affect execution of instructions

- Simple logic (PC) reacts faster than complex logic (IU)
- Send two clock pulses in the space of one
- PC advances to skip an instruction



```
      set ctr = 8
loop: read PIN digit
      cmp stored PIN digit
      jne error
      loop ctr
```

Attacks on Smart Cards (ctd)

Erasing an EEPROM cell requires a high voltage (12 vs. 5V) charge

- Don’t provide the power to erase cells
- Most cards now generate the voltage internally
  - Destroy the (usually large) on-chip voltage generator to ensure that the memory is never erased

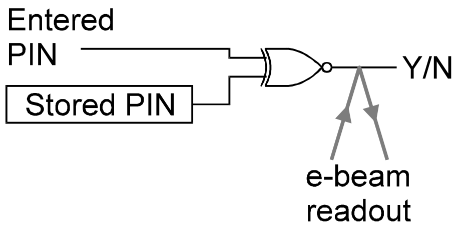
Physical Attacks

Erase onboard EPROM with UV spot beam

Remove chip from encapsulation with nitric acid

- Use microprobing to access internal circuit sections
- Use an electron-beam tester to read signals from the operational circuit

Example: PIN recovery with an e-beam tester



Physical Attacks (ctd)

Modify the circuit using a focused ion beam (FIB) workstation

- Disable/bypass security circuitry (Mondex)
- Disconnect all but EEPROM and CPU read circuitry

Use radiation to imprint the device state

- CMOS devices lock up in their current state

Data retention time is temperature-dependent

- At low temperatures, devices will retain state from seconds to minutes after power is removed
- Liquid-helium attack: At absolute zero, device becomes superconducting and retains charge more or less indefinitely

Physical Attacks (ctd)

Circuit operation changes at elevated temperatures

- Hot carriers tunnel into the substrate, leaving a latent image of data patterns
- Electromigration physically moves metal atoms based on current flow
  - This is a standard device failure mechanism at elevated temperatures

Attacking the Random Number Generator

Generating good random data (for encryption keys) on a card is exceedingly difficult

- Self-contained, sealed environment contains very little unpredictable state

Possible attacks

- Cycle the RNG until the EEPROM locks up
- Drop the operating voltage to upset analog-circuit RNGs
- French government attack: Force manufacturers to disable key generation
  - This was probably a blessing in disguise, since externally generated keys may be much safer to use

Timing/Power Analysis

Crypto operations in cards

- Take variable amounts of time depending on key and data bits
- Use variable amounts of power depending on key and data bits
  - Transistors are voltage-controlled switches that consume power and produce electromagnetic radiation
  - Power analysis can provide a picture of DES or RSA en/decrypt operations
  - Recovers 512-bit RSA key at ~3 bits/min on a PPro 200

Differential power analysis is even more powerful

- Many card challenge/response protocols are DES-based → apply many challenge/response operations and observe power signature