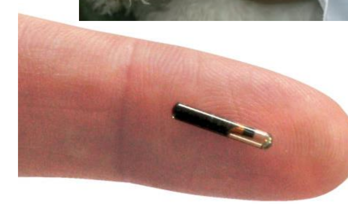
Smartcards

Erik Poll
Digital Security
Radboud University Nijmegen

Applications of smartcards & RFID tags

"Payment"



"ID"

Exit smart cards?

mobile payments

mDL
(mobile Driving License)

This may use secure hardware inside the mobile phone:
Apple Secure Enclave or Android hardware-backed keystore

Why use smartcards?

Why use smartcards?

- ***What are the security objectives?***
- ***What are the capabilities of the smartcard that let it realise these objectives?***
- ***How are these security objectives guaranteed?***
- ***What are the security assumptions all this relies on?***

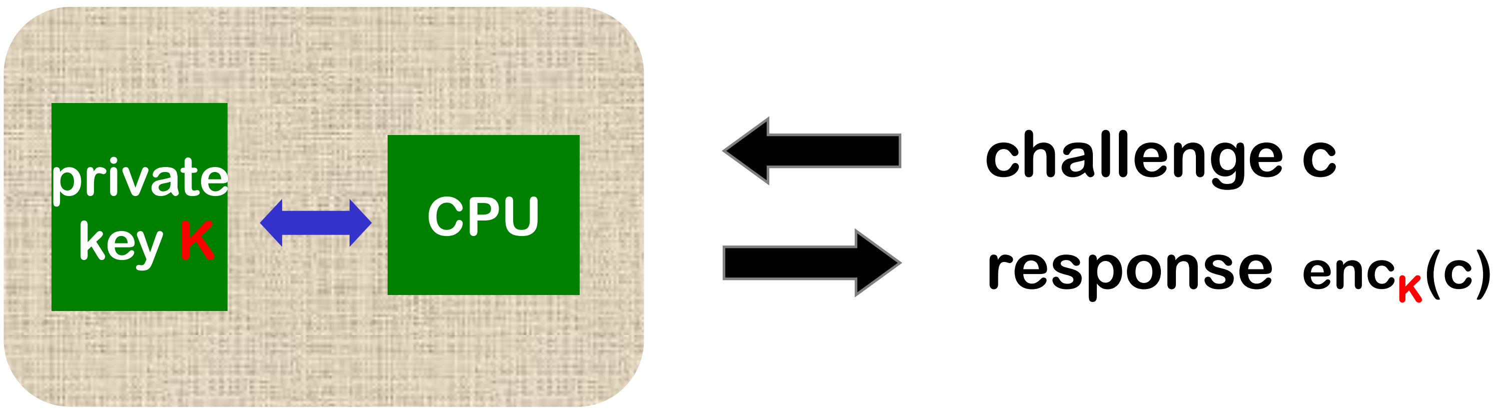
Security objectives

- **Identification** of the card and/or the card holder

- **Authentication** of the card and/or the card holder
- **Non-repudiation** of some action
  - In Dutch: onweerlegbaarheid

- **Integrity** – of the software & data on the card
- **Confidentiality** – esp. of the data on the card

How to achieve authentication or non-repudiation



- *If* card can perform encryption,
  then the private key K *never* has to leave the card
- The card issuer does not have to trust the network, the terminal, or the card holder
- The card can also sign a message using asymmetric crypto, or compute a MAC using symmetric crypto.

Security (critical) functionality in smartcard

- Crypto: storing cryptographic keys &
  executing cryptographic operations

- Access control for functionality
  - Eg with a PIN code
  - Incl. functionality to install keys!
    Easy to overlook, but crucial of course…

Crypto solves some problems

- ensuring integrity, authenticity, non-repudiation, confidentiality,…

but also introduces new problems:

- Where to store keys?
- How to distribute them?
- What hw/sw can we trust to do crypto operations?
- How to ensure integrity & confidentiality
  of the cryptographic key?
  Here we will need access control again

Overview of today

- What is a smartcard?
- Hardware, protocols

What is a smartcard?

- Tamper-resistant computer, embedded in piece of plastic, with limited resources
- aka *chip card* or *integrated circuit card (ICC)*

- capable of securely
  - storing data
  - processing data
    - This is what makes a smartcard *smart*; stupid cards can store but not process
    - Processing capabilities vary greatly!

Smartcard form factors

- **traditional** credit-card sized plastic card
  - ISO 7816

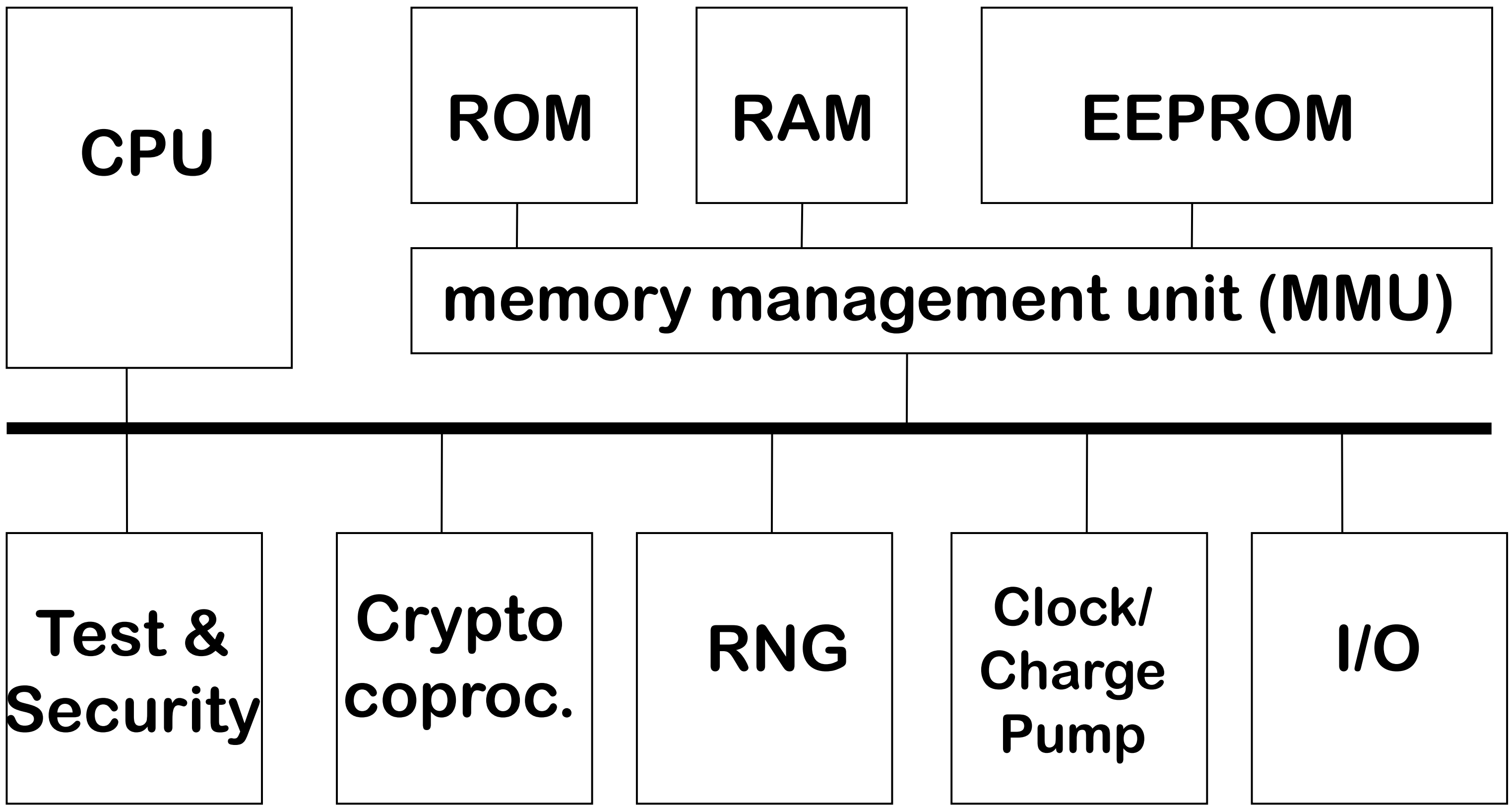
- mobile phone **SIM**
  - cut-down in size

- **contactless** cards
  - aka *proximity card* or *RFID tag*
  - also possible: dual interface

Stupid vs smart smartcards

- Memory cards ("stupid" smartcards)
  - essentially just provide a file system
  - possibly with some access control or, simpler still, destructive (irreversible) writes as in old payphone-cards
  - configurable functionality hardwired in ROM
- Microprocessor cards ("smart" smartcard)
  - contain CPU
    - possibly also crypto co-processor
  - programmable
    - program burnt into ROM, or stored in EEPROM
- RFID tags are often like old memory cards
- Focus in this course will be on microprocessor cards

Smart card hardware



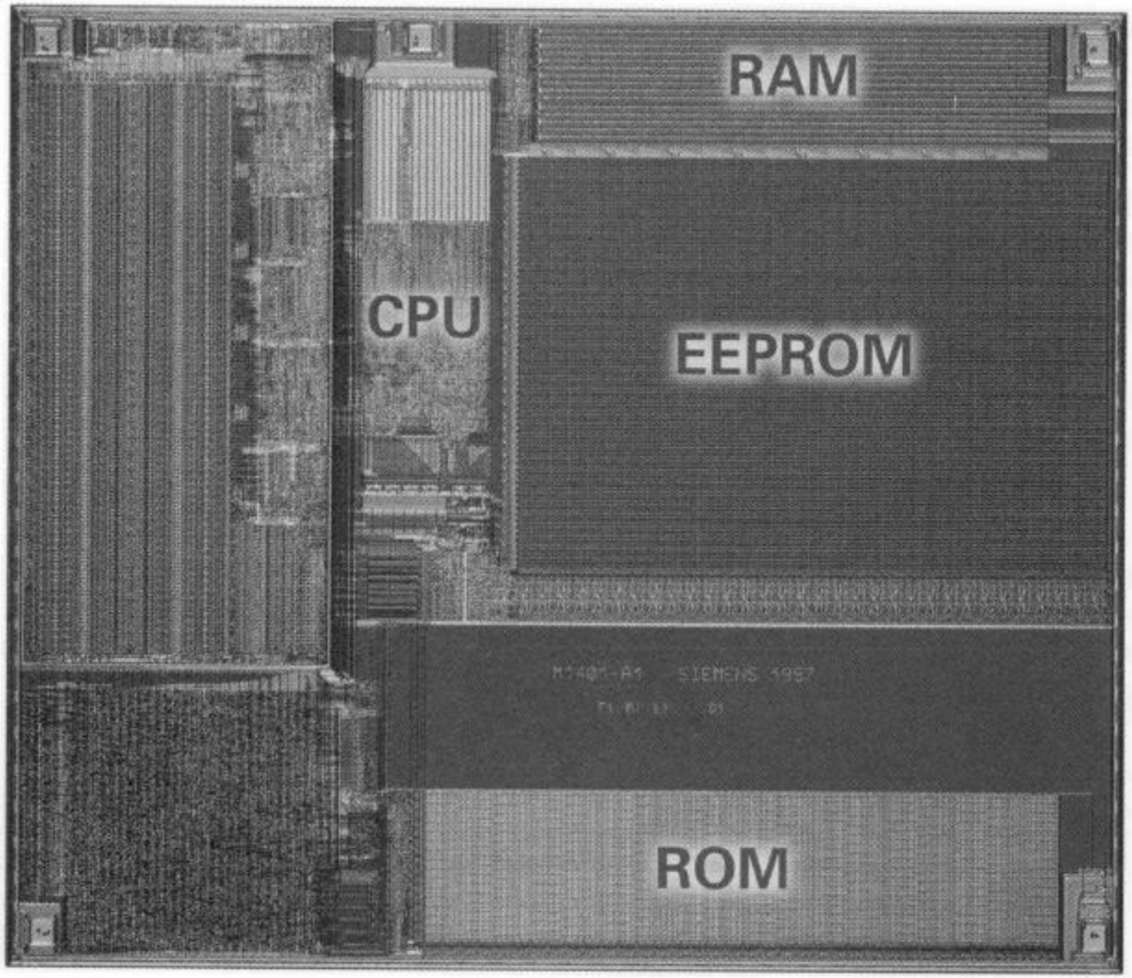


Figure 3.38 Photo of an SLE 66CX160S Smart Card microcontroller with an area of 21 mm^2. This chip was made using 0.6-µm technology and has 32 kB of ROM, a 16-kB EEPROM and 1280 bytes of RAM. The two unlabeled regions on the left-hand side of the chip are the numeric coprocessor and the peripheral elements (timer, random-number generator and CRC arithmetic processor). The five bonding pads for the electrical connections to the module contacts can be clearly seen in the photo.

Smartcard hardware

- CPU
- memory
  - volatile (RAM) and non-volatile (ROM+EEPROM)
- limited I/O: just a serial port

Possibly:

- crypto co-processor
- random number generator (RNG)

No clock, no power!

CPU

- 8, 16 and now also 32 bit
- Steady need for more processing powers
  - for virtual machines & cryptography

Crypographic co-processor

- **DES, AES**
  - **DES in hardware takes same size as DES program code in ROM**

- **For public-key crypto, ALU providing exponentation and modulo arithmetic with large numbers**

Smartcard memory

ROM

- BIOS + operating system

EEPROM

- the smartcard's hard disk

RAM

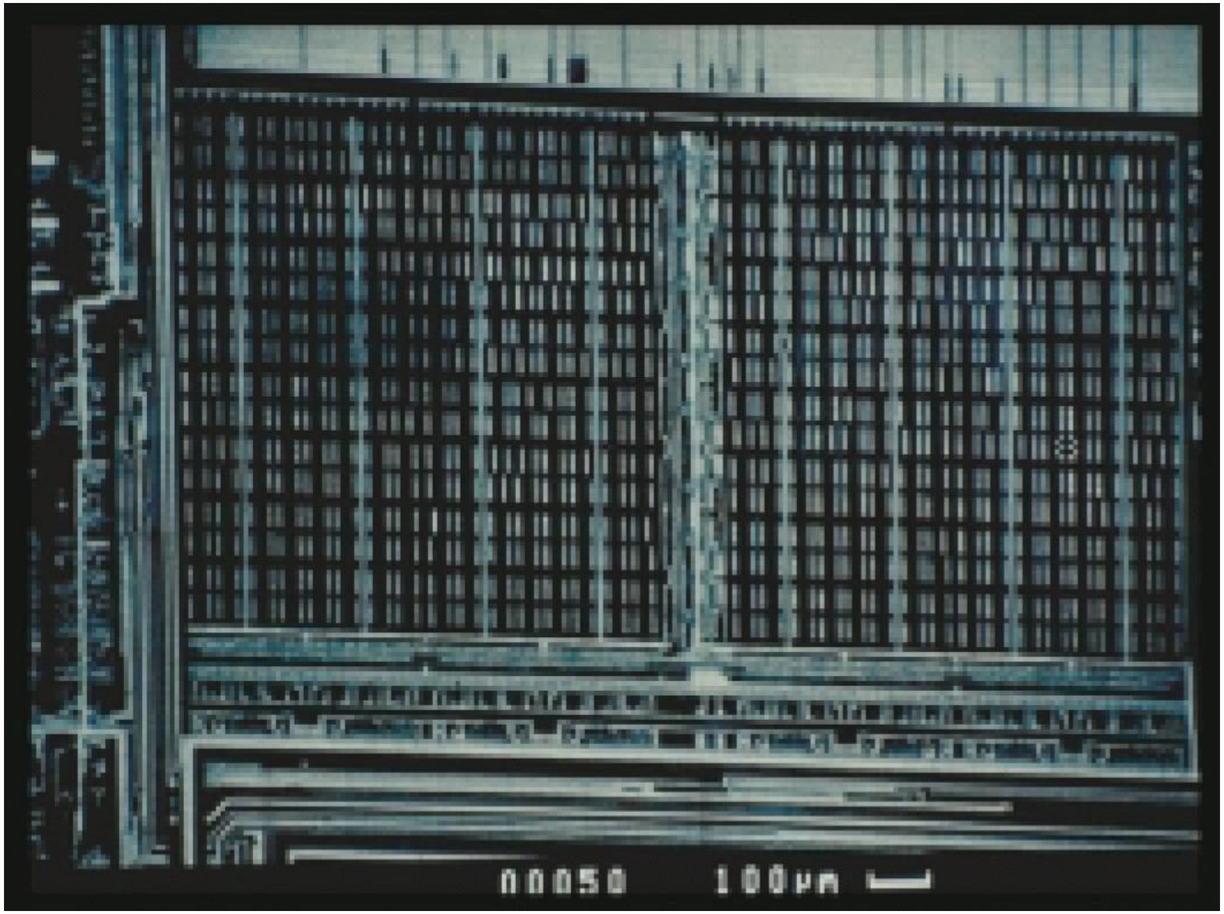
- workspace

Typical modern card may have 512 bytes RAM, 16K ROM, 64K EEPROM, 13.5 MHz

RAM

- **Volatile aka transient memory**
- **SRAM (static RAM) used rather than DRAM (Dynamic RAM) for lower power consumption**
- **Used for temporary data**
  - **stack**
  - **I/O buffer**
- ***Typically 128 bytes to 16 Kbyte***
- **Volatile, but small permanent storage characteristics**

Reading RAM with scanning electron microscope?



Very tricky, and only if card operates at a low frequency

ROM

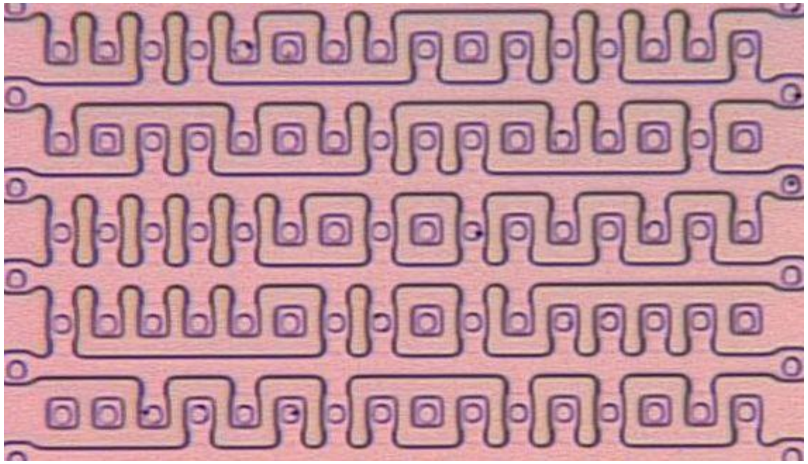
- Permanent storage
- Filled during production, using ROM mask
- Contains OS + possibly application code

- *Typically 8 to 512 Kbyte*

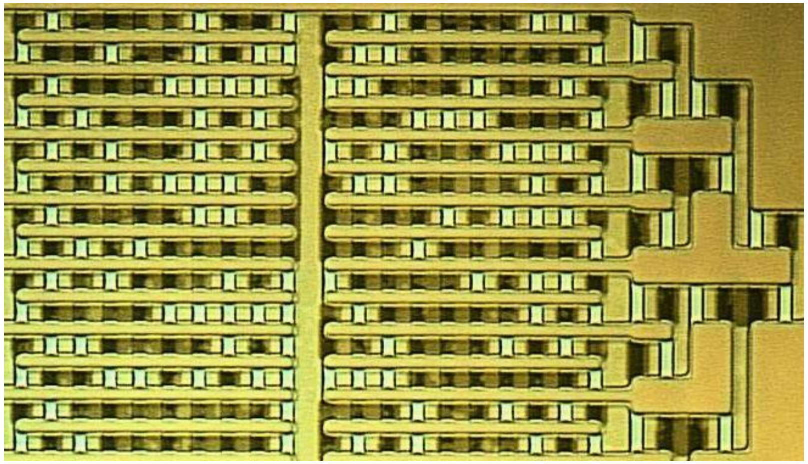
- Flash is coming up as replacement of ROM

- Optically readable after removing top layers

Extraction of ROM content



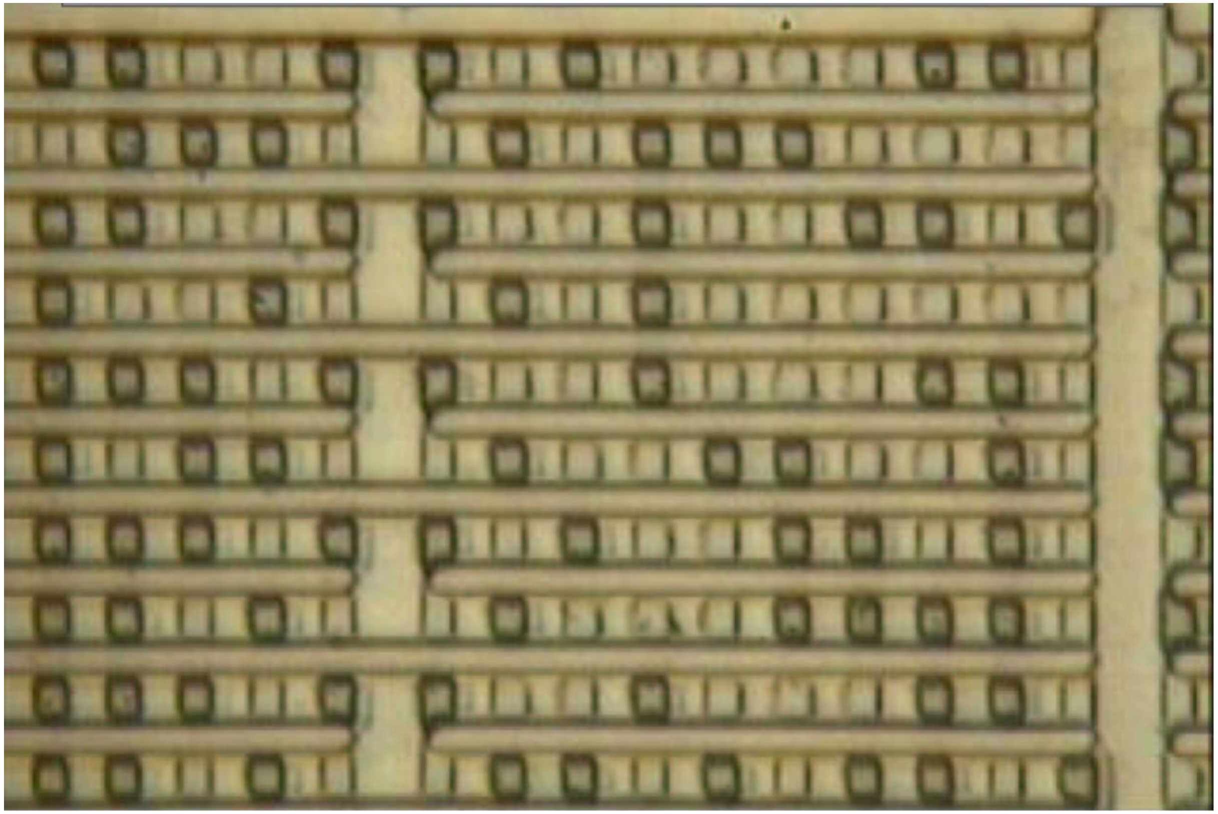
10x16 bits of NOR ROM
visible outlines reveal content



14x20 bits of NAND ROM
after staining, shadows reveal content

[Source: Oliver Kömmerling, Marcus Kuhn]

Staining for ion implant ROM array



[Source: Brightsight]

EEPROM

- Non-volatile aka persistent, rewritable memory
- Used for applications and data:
  - "the smartcard's hard disk“

- *Typically 1 to 512 Kbyte*

- Characteristics:
  - slow: 1000x slower than RAM
  - has limited lifetime in # writes (but > 10^6)
  - writing is two stage operation: erase & write
  - data retention 10-100 years
  - writing takes high power consumption
    - 17V, realised using charge pump

ROM vs EEPROM for program code?

Choice between code in ROM or EEPROM has big impact on development

- Program code in ROM must be suitable for mass-use
  - And can *never* be patched or updated
- Trend away from having a lot of code in ROM towards
  - eg just some library code or OS in ROM,
    and all custom functionality (the 'application') in EEPROM

Other non-volatile memory types

Modern alternatives for EEPROM and ROM

- Flash
  - writing 10 μs instead of 3 -10 ms for EEPROM
  - programming voltage 12V, against 17V for EEPROM
  - >100,000 writes, >10 years data retention
  - Flash replacing ROM eliminates need for ROM mask & reduces development time

- FRAM (Ferro-electric RAM)
  - writing 100 ns
  - programming voltage 5V

Size matters!

Memory types also vary in amount of chip surface per byte:

RAM requires more space than EEPROM,
EEPROM requires more space than ROM

Size of chip is a major cost factor, hence little RAM

Size = Money

Comparison of memory types

	# of write/ erase cycles	writing time	typical size
RAM	unlimited	70 ns	1700 μm^2
EEPROM	$>10^4$-10^6	3-10 ms	400 μm^2
Flash	$>10^5$	10 μs	200 μm^2
FRAM	$>10^{10}$	100 ns	200 μm^2
ROM	-	-	100 μm^2

source: W. Rankl & W. Effling, Smartcard Handbook, 2nd edition, 2000

These numbers give a rough impression, but are outdated !!!

Memory Management

- Responsible for allocating memory

- Possibly also for some memory access control, eg
  - between the different types of memory (eg stack, program data, and code)
  - between application code and the operating system (OS)
  - between applications, for cards that allow multiple applications to be installed

Clock circuit & charge pump

- Charge pump
  - to generate EEPROM programming voltage
- Clock circuit
  - eg. division of the external clock signal to get a lower clock frequency for I/O

- Typical clock speed 8-13.5 MHz

- Esp. for SIM cards: the processor can go into sleep-mode, where clock pulse is stopped, to reduce power consumption

Test & Security

- Self-test hardware & software
  - checking if all RAM & EEPROM cells work
  - checksums for ROM and static EEPROM

- Possible additional monitoring and response against attacks

- Test functionality is a security risk and may partly have to be disabled before personalisation!
  - by writing EEPROM, blowing fuses, or even physically removing hardware

Random Number Generation (RNG)

- **Needed for key generation & fresh nonces**
- **Typically pseudo RNG (PRNG) in software**
  - True RNG that is immune to external influence is hard to realise in hardware
- **NB different cards of same production batch should produce *different* sequences of random numbers**
  - PRNG using seed (stored in EEPROM) supplied as part of OS initialisation

- ***Potential hardware security problem with storing PNRG state in EEPROM?***

What if this EEPROM wears out after generating lots of random numbers? Card may have to check for this !!

NB nonces vs randoms

Many security protocols require the use of nonces (numbers only used once)

- to prevent replay attacks (aka ensure freshness)

- Random numbers are only one of the ways to generate nonces

- Another way is to simply have an increasing counter
  - Eg, your bank cards all use have a counter that is included in integrity & non-repudiation checks
  - Wrapping around after an integer overflow is a security risk for such counters

Random Number Generation (RNG)

Tested eg according to FIPS 140-1 which states single stream of 20,000 consecutive bits must pass

- monobit test: 9,546 < # ones < 10,346
- poker test: frequency of 4 bits patterns:
  - divide in 5000 4 bits segments,
  - calculate f(p) = # of occurrence of 4 bits pattern p
  - $1.03 < X < 57.4$, where X is $16/5000 * \Sigma_p f(p)^2 - 5000$
- run test: requirements of runs (sequence of all 0's or all 1's) of lengths 1-6
- long run test: no runs ≥ 34