Smart Card Tutorial - Part 1

First Published in September 1992

Introduction To Smart Cards

Even the name Smart Card captures the imagination, however such a term is ambiguous and is used in many different ways. ISO uses the term, Integrated Circuit Card (ICC) to encompass all those devices where an integrated circuit is contained within an ISO 1 identification card piece of plastic. The card is 85.6mm x 53.98mm x 0.76mm and is the same as the ubiquitous bank card with its magnetic stripe that is used as the payment instrument for numerous financial schemes.

Integrated Circuit Cards come in two forms, contact and contactless. The former is easy to identify because of its gold connector plate (fig 1). Although the ISO Standard (7816-2) defined eight contacts, only 6 are actually used to communicate with the outside World. The Contactless card may contain its own battery, particulary in the case of a "Super Smart Card" which has an integrated keyboard and LCD display. In general however the operating power is supplied to the contactless card electronics by an inductive loop using low frequency electronic magnetic radiation. The communications signal may be transmitted in a similar way or can use capacitive coupling or even an optical connection.

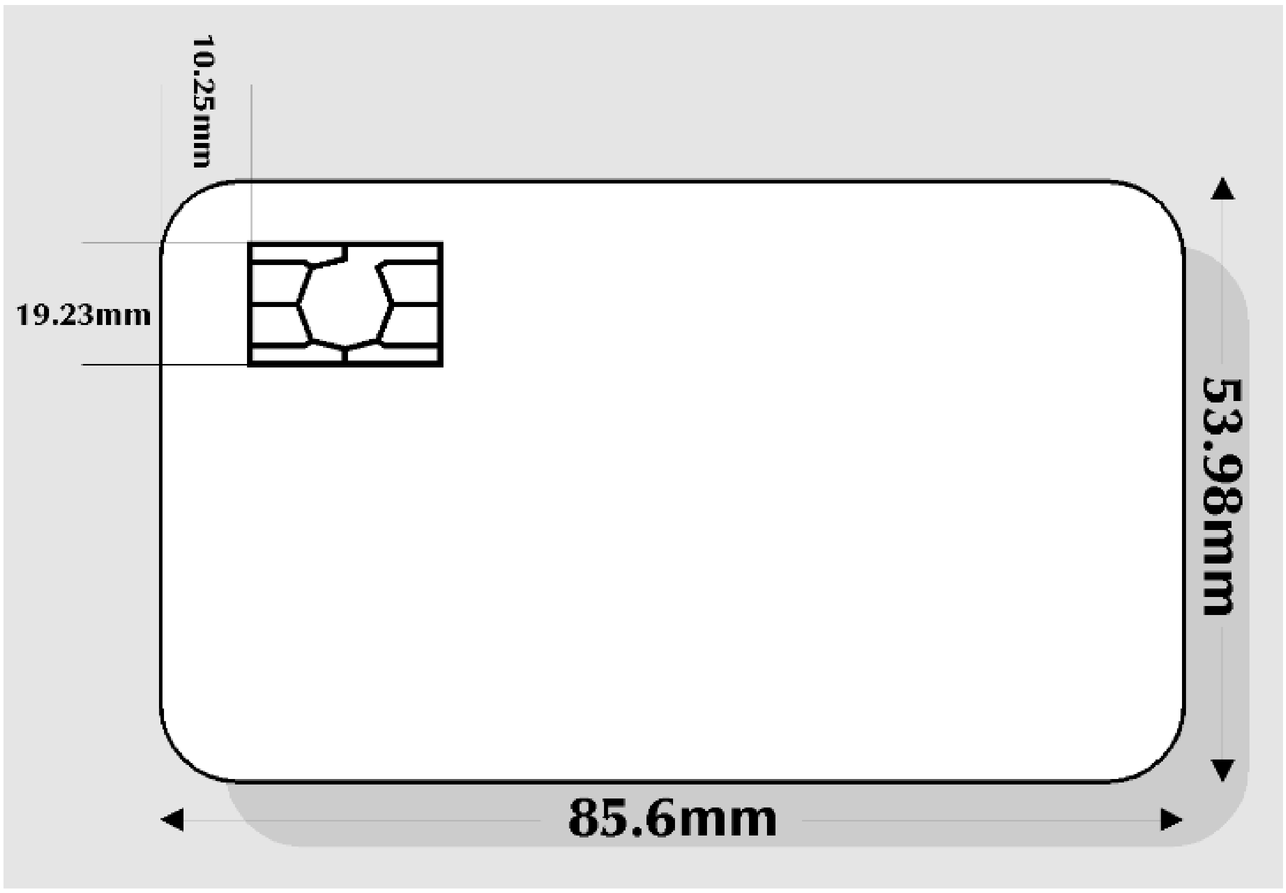


Figure 1: ISO ID 1Card

The Contact Card is the most commonly seen ICC to date largely because of its use in France and now other parts of Europe as a telephone prepayment card.. Most contact cards contain a simple integrated circuit although various experiments have taken place using two chips. The chip itself varies considerably between different manufacturers and for a whole gambit of applications. Let us consider first the purpose for the 6 contacts used by the ICC (fig 2)

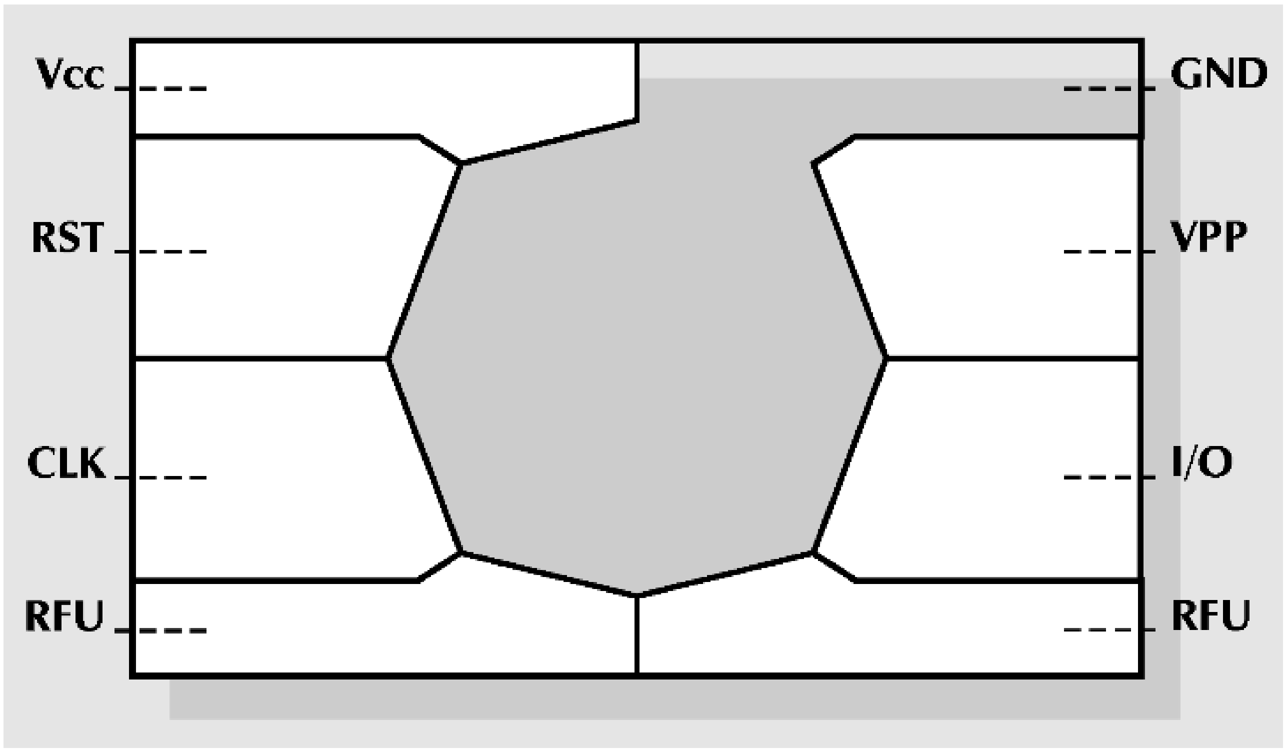


Figure 2: ISO 7816-2 Connector

Vcc is the supply voltage that drives the chips and is generally 5 volts. It should be noted however that in the future we are likely to see a move towards 3 volts taking advantage of advanced semiconductor technology and allowing much lower current levels to be consumed by the integrated circuit. Vss is the substrate or ground reference voltage against which the Vcc potential is measured. Reset is the signal line that is used to initiate the state of the integrated circuit after power on.This is in itself an integral and complex process that we shall describe later in more detail.

The clock signal is used drive the logic of the IC and is also used as the reference for the serial communications link. There are two commonly used clock speeds 3.57 MHZ and 4.92 MHZ. The lower speed is most commonly used to date in Europe but this may change in the future. One may be tempted to ask why these strange frequencies were chosen, why not just a straight 5 MHZ. The reason lies in the availability of cheap crystals to form the clock oscillator circuits. Both of these frequencies are used in the television world for the colour sub carrier frequency. The PAL system operates using 4.92 MHZ whilst the 3.57 MHZ is used by the American NTSC standard. The the Vpp connector is used for the high voltage signal that is necessary to program the EPROM memory. Last, but by no means least is the serial input/output (SIO) connector. This is the signal line by which the chip receives commands and interchanges data with the outside world. This is also a fairly complex operation and will be the subject of a more detailed discussion where symbols such as T0 and T1 will be fully explained.

So what does the chip contain, well the primary use of the IC card is for the portable storage and retrieval of data. Hence the fundamental component of the IC is a memory module. The following list represents the more commonly used memory types,

■ ROM	Read only memory (mask ROM)
■ PROM	Programmable read only memory
■ EPROM	Erasable programmable ROM
■ EEPROM	Electrically erasable PROM
■ RAM	Random access memory

A particular chip may have one or more of these memory types. These memory types have particular characteristics that control their method of use. The ROM type of memory is fixed and can not be changed once manufactured by the semiconductor company. This is a low cost memory, in that, it occupies minimum space on the silicon substrate. The use of the silicon is often referred to as real estate because clearly one wants to get as much as possible into the smallest possible space. The snag however is that it can not be changed and takes several months to be produced by the semiconductor company. There is also effectively a minimum order quantity in order to achieve this low cost.

In order of increasing real estate the PROM comes next. This memory is programmable by the user through the use of fusible links. However high voltage and currents are required for the programming cycle and such devices are not normally used in Integrated Circuit Cards. The EPROM has been widely used in the past but the name for this application is something of a misnomer. Whilst the memory is erasable, by means of ultra violet light, the necessary quartz window is never available in the ICC and the memory is really used in one time programmable mode (OTP). Getting pretty heavy in real estate terms is the EEPROM. This memory is indeed erasable by the user and can be rewritten many times (between 10,000 and 1,000,000 in a typical implementation) All of these memories describe so far are non volatile. In other words when the power is removed they still retain their contents. The random access memory (RAM) is a different kettle of fish, this is volatile memory and as soon as the power is removed the data contents is lost.

In order to pursue our studies further we must note that the cost of the IC at saturation (i.e when development costs have been recouped) is proportional to the square area of silicon used (assuming constant yield). The ISO connector is so designed to constrain the silicon die size to about 25mm^2 (although it is possible to handle 35mm^2 or more). However the important point is more concerned with reliability where clearly the larger die will be more prone to mechanical fracture. There is another bi- product that we will consider later where the cost of testing and personalisation are considerable altered by the complexity of the particular chip. It is clear however that we should attempt to minimise the contents of the chip on both cost and reliability grounds commensurate with the particular application .

Well of course you can't have something for nothing and although a telephone card may operate with a little EEPROM memory (128 - 512 bytes) and the memory control logic, more sophisticated applications will demand ROM, EEPROM, RAM and a CPU (Central Processing Unit) to achieve the necessary business. It is the addition of the CPU or micro-controller that really leads to the term "Smart" although we will not be rigorous in our use of the term.

The control logic should not be overlooked as this is necessary not only for communication protocols but also to offer some protection of the memory against fraudulent use. The ICC is probably the security man's dream because unlike most electronic storage and processing devices it has security intrinsically built in. The ICC really does provide a tamper resistant domain that is difficult to match with the some what larger security boxes that handle cryptographic processes.

So now we can differentiate the different types of ICC by their content,

- Memory only
- Memory with security logic
- Memory with CPU

The security logic can be used to control access to the memory for authorized use only. This is usually accomplished by some form of access code which may be quite large (64 bits or more). Clearly the use of EEPROM memory must be strictly controlled where fraudsters can obtain a financial advantage by unauthorized use. This applies as much to telephone cards as applications using ICC for cryptographic key carriers. The security advantage of the CPU device is of course more significant because the CPU is capable of implementing cryptographic algorithms in its own right, but we will discuss this in more detail in due course.

In the Smart Card world the term application is widely used to describe the software or programs that the IC implements. In the simplest case the application may be just a file manager for organising the storage and retrieval of data. Such an application may be totally implemented in the logic of the chip. Similarly the chip must contain the communications logic by which it accepts commands from the card acceptance device (CAD) and through which it receives and transmits the application data. The ICC which contains a CPU can handle more sophisticated applications and even multi applications since the CPU is also capable of processing the data and taking decisions upon the various actions that may be invoked. The subject of mult applications and particulary the implementation of security segregation is another subject for more detailed discussion in subsequent parts.

Smart Card Tutorial - Part 2

First Published in October 1992

How the IC card is made

The manufacture of a smart card involves a large number of processes of which the embedding of the chip into the plastic card is key in achieving an overall quality product. This latter process is usually referred to as card fabrication. The whole operation starts with the application requirements specification. From the requirements individual specifications can be prepared for the chip, card, mask ROM software and the application software. The ROM software is provided to the semiconductor supplier who manufactures the chips. The card fabricator embeds the chip in the plastic card. It is also quite normal for the fabricator to load the application software and personalisation data. Security is a fundamental aspect in the manufacture of a smart card and is intrinsic to the total process. However we will consider security separately in subsequent articles in this series. We will look at each of the stages in the manufacture of the smart card as shown in figure. 1.

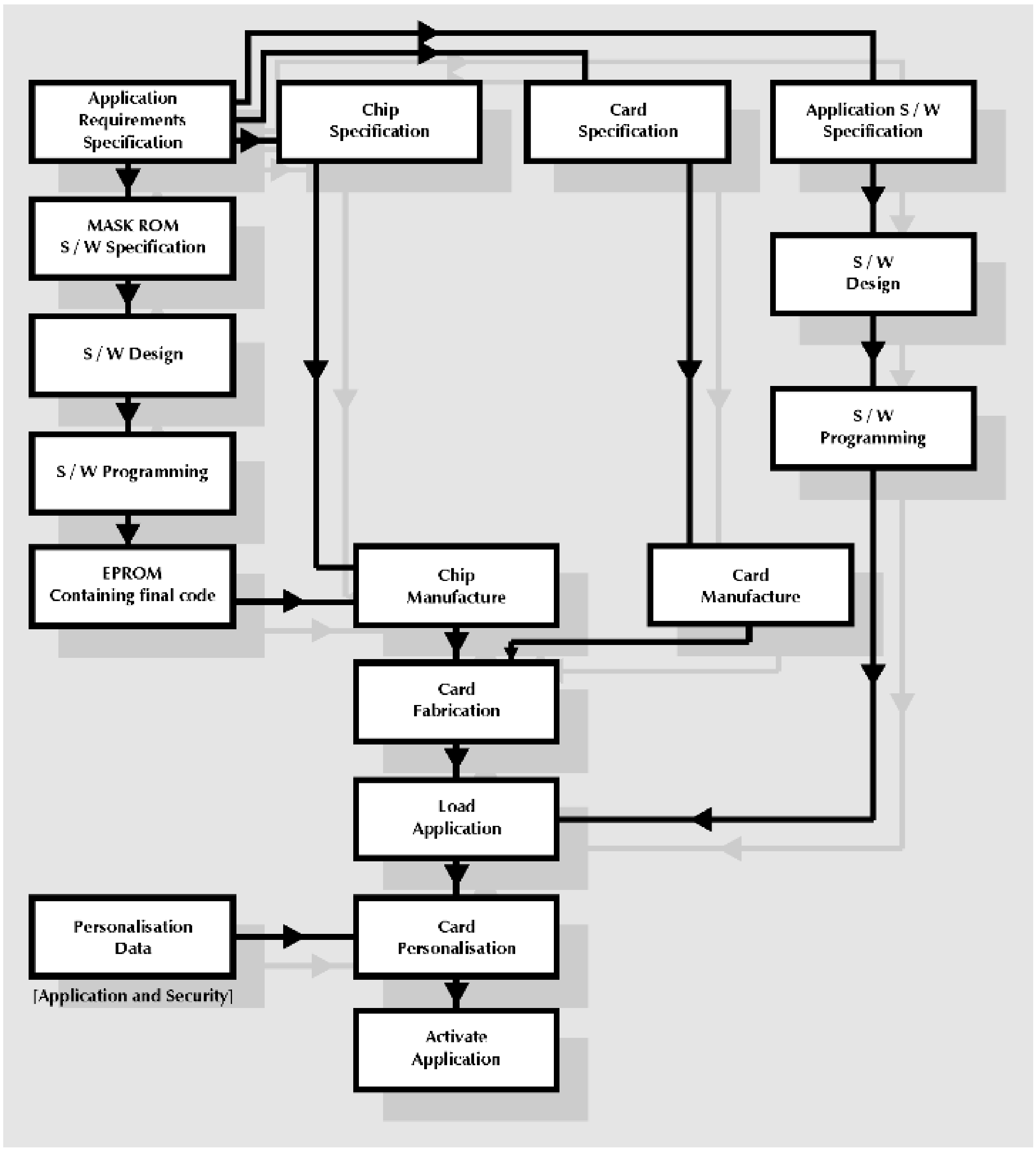


Figure 1: Stages in the manufacture of a Smart Card

Chip specification

There are a number of factors to be decided in the specification of the integrated circuit for the smart card. For the purpose of this discussion we will consider a CPU based card although the manufacture of a memory card is substantially a subset of that described here. The key parameters for the chip specification are as follows,

- Microcontroller type (e.g 6805,8051)
- Mask ROM size
- RAM size

- Non volatile memory type (e.g EPROM, EEPROM)
- Non volatile memory size
- Clock speed (external, and optionally internal)
- Electrical parameters (voltage and current)
- Communications parameters (asynchronous, synchronous, byte, block)
- Reset mechanism
- Sleep mode (low current standby operation)
- Co-processor (e.g for public key cryptography)

In practice the semiconductor manufacturers have a range of products for which the above parameters are pre-defined. The task of the designer is therefore concerned with choosing the appropriate product for the particular application. As mentioned previously security may be an important issue for the application and accordingly there may be extra requirements on the physical and logical security offered by the particular chip. Conformance to ISO standards is also likely to be a requirement and in this area ISO 7816 - 3 (Electronic signals and transmission protocols) is the principle standard to be considered. It should be noted however that ETSI (European Telecommunications Standard Institute) are currently developing new standards for the CEN TC224 committee. These standards are more stringent than that described by the ISO standards. For example the ISO 7816-3 allows a card current supply of up to 200 mA. ETSI have recommended 20mA for normal use and 10mA for applications such as portable phones.

Card specification

The specification of a card involves parameters that are common to many existing applications using the ISO ID-1 card. The following list defines the main parameters that should be defined,

- Card dimensions
- Chip location (contact card)
- Card material (e.g PVC, ABS)
- Printing requirements
- Magnetic stripe (optional)
- Signature strip (optional)
- Hologram or photo (optional)
- Embossing (optional)
- Environmental parameters

The characteristics of the smart card are part of the ISO 7816 part 1 (physical) and 2 (contact location) standards. The choice of chip location has been a difficult subject due largely to the use of magnetic stripes. The early French cards put the IC module further off the longitudinal axis of the card than the standard eventually agreed by ISO.

This was preferable because of the residual risk of chip damage due to bending. The French Transac tracks were lower on the card which also made this position preferable. The now agreed ISO standards for magnetic stripes resulted in the French chip position and the magnetic stripe being coincident. Hence the now agreed lower location which does of course result in higher bending stress on the chip. The ISO 7816-2 standard does however allow the position of the contacts to be either side of the card. More recently there have been moves to remove this option with the front (opposite to the side containing the magnetic stripe) being the preferred position for the IC connector.

The choice of card material effects the environmental properties of the finished product. PVC was traditionally used in the manufacture of cards and enabled a higher printing resolution. Such cards are laminated as three layers with transparent overlays on the front and back. More recently ABS has been used which allows the card to be produced by an injection moulding process. It is even proposed that the chip micromodule could be inserted in one step as part of the moulding process. Temperature stability is clearly important for some applications and ETSI are particulary concerned here, such that their higher temperature requirement will need the use of polycarbonate materials.

Mask ROM Specification

The mask ROM contains the operating system of the smart card. It is largely concerned with the management of data files but it may optionally involve additional features such as cryptographic algorithms (e.g DES). In some ways this is still a relatively immature part of the smart card standards since the early applications used the smart card largely as a data store with some simple security features such as PIN checking. The relevant part of the ISO standard is 7816-4 (commands). There is a school of thought that envisages substantial changes in this area to account for the needs of multi-application cards where it is essential to provide the necessary security segregation. The developed code is given to the supplier who incorporates this data as part of the chip manufacturing process.

Application Software Specification

This part of the card development process is clearly specific to the particular application. The application code could be designed as part of the mask ROM code but the more modern approach is to design the application software to operate from the PROM non volatile memory. This allows a far more flexible approach since the application can be loaded into the chip after manufacture. More over by the use of EEPROM it is possible to change this code in an development environment. The manufacturer of a chip with the users ROM code takes on average three months. Application code can be loaded into the PROM memory in minutes with no further reference to the chip manufacturer.

Chip Fabrication

The fabrication of the card involves a number of processes as shown in fig. 2. The first part of the process is to manufacture a substrate which contains the chip. This is often called a COB (Chip On Board) and consists of a glass epoxy connector board on which the chip is bonded to the connectors. There are three technologies available for this process, wire bonding, flip chip processing and tape automated bonding (TAB). In each case the semiconductor wafer manufactured by the semiconductor supplier is diced into individual chips . This may be done by scribing with a diamond tipped point and then pressure rolling the wafers so that it fractures along the scribe lines. More commonly the die are separated from the wafer by the use of a diamond saw. A mylar sheet is stuck to the back of the wafer so that following separation the dice remain attached to the mylar film.

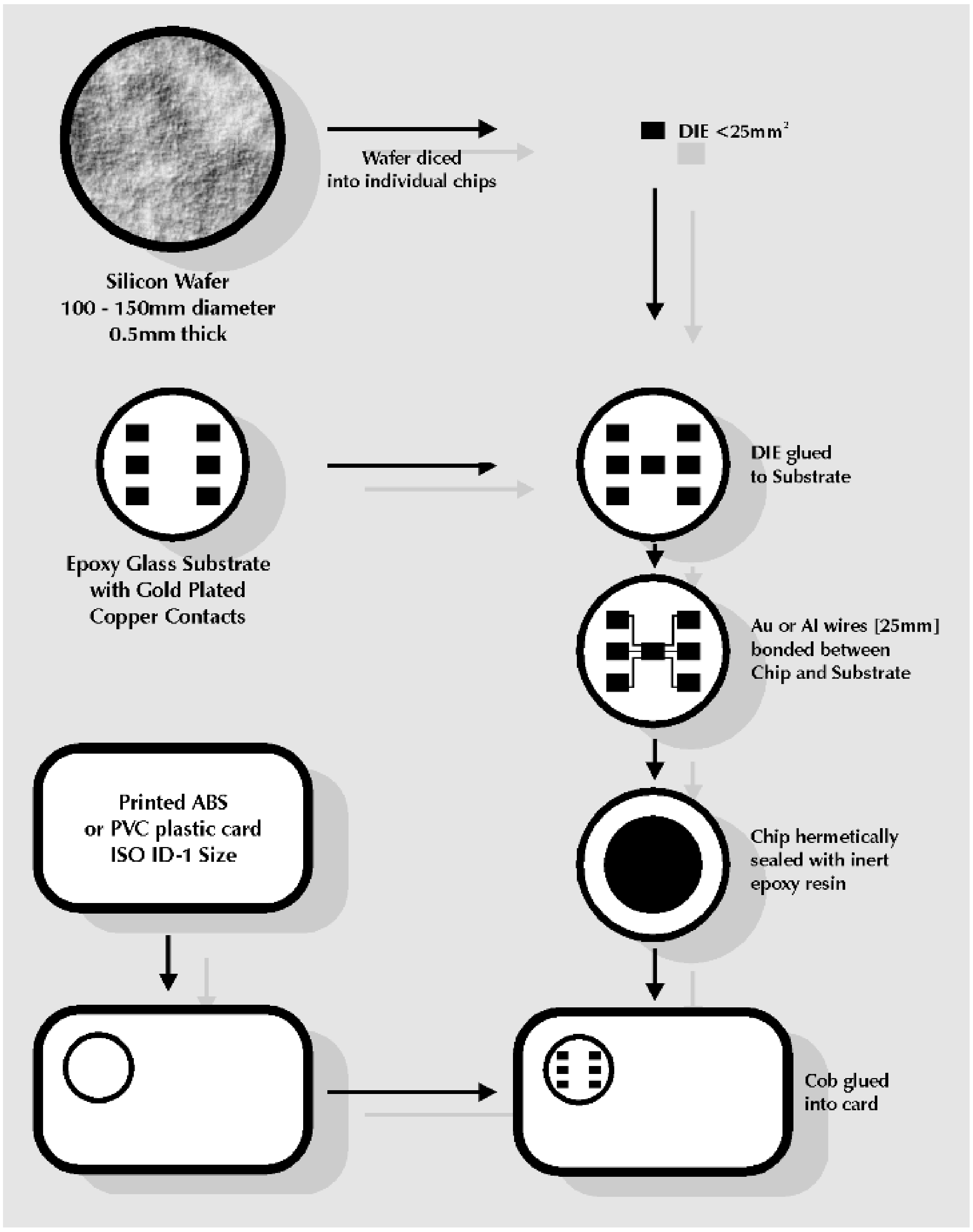


Figure 2: Smart Card fabrication process

Wire bonding is the most commonly used technique in the manufacture of smart cards. Here a 25uM gold or aluminium wire is bonded to the pads on the chip using ultrasonic or thermo compression bonding. Thermo compression bonding requires the substrate to be maintained at between 150C and 200C. The temperature at the bonding interface can reach 350C. To alleviate these problems thermo sonic bonding is often used which is a combination of the two processes but which operate at lower temperatures.

The die mounting and wire bonding processes involve a large number of operations and are therefore quite expensive. Because in general only 5 or 6 wires are bonded for smart card applications this approach is acceptable. However in the semiconductor industry generally two other techniques are used, the flip chip process and tape automated bonding. In both cases gold bumps are formed on the die. In flip chip processing the dice are placed face down on the substrate and bonding is effected by solder reflow. With tape automated bonding the dice are attached by thermocompression to copper leads supported on a flexible tape similar to a 35mm film.

The finished substrate is hermetically sealed with an inert material such as epoxy resin. The complete micromodule is then glued into the card which contains the appropriately sized hole.

The fabrication of a contactless card is somewhat different since it always involves a laminated card as shown in fig. 3. The ICs and their interconnections as well as the aerial circuits are prepared on a flexible polyimide substrate.

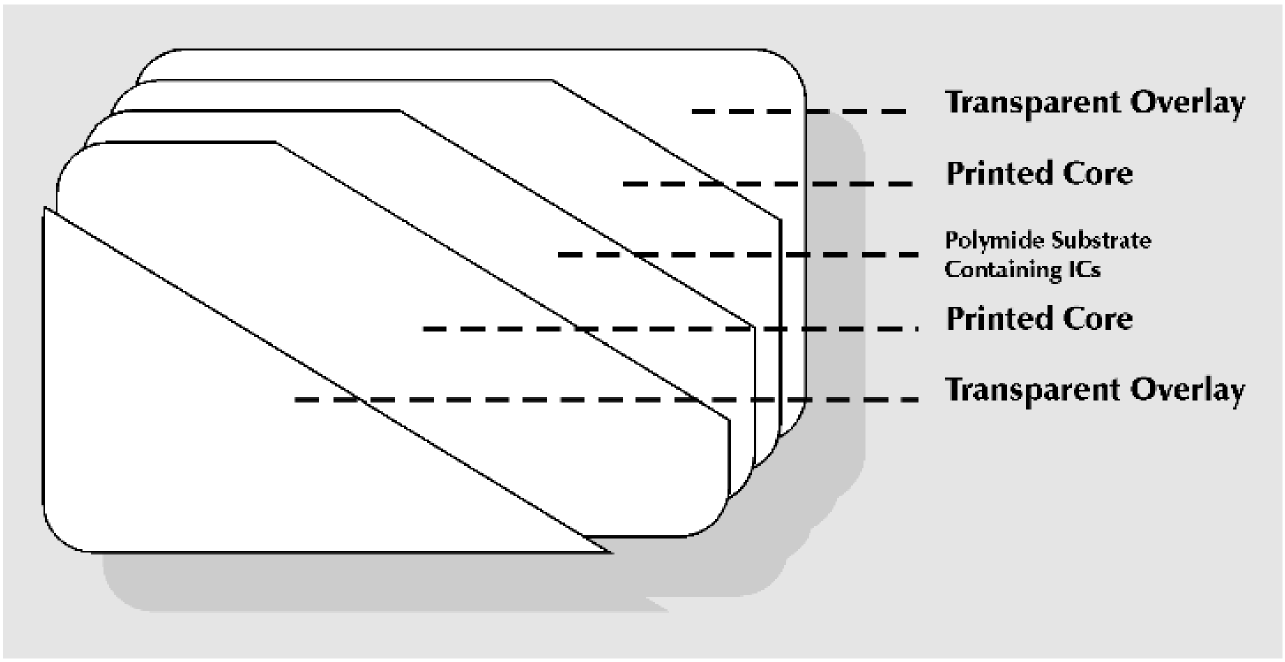


Figure 3: Contactless card laminations

Application load

Assuming the application is to be placed in the PROM memory of the IC then the next stage in the process is to load the code into the memory.

This is accomplished by using the basic commands contained in the operating system in the mask ROM. These commands allow the reading and writing of the PROM memory.

Card Personalisation

The card is personalised to the particular user by loading data into files in the PROM memory in the same way that the application code is loaded into memory. At this stage the security keys will probably be loaded into the PROM memory but as mentioned previously we will explore this in more detail later.

Application Activation

The final operation in the manufacturing process is to enable the application for operation. This will involve the setting of flags in the PROM memory that will inhibit any further changes to be made to the PROM memory except under direct control of the application. Again this is an integral part of the overall security process.

Smart Card Tutorial - Part 3

First Published in November 1992

Physical characteristics of the Contact Card

Many observers have commented that the widespread use of smart cards is being impeded by the lack of standards. Interoperability is of course the name of the game and is the primary purpose of standards. The problems of interoperability start at the bottom, in other words with the physical dimensions of the card and the location of the contacts.

These standards are well established and as we shall show in subsequent parts so are the more important characteristics of a smart card that form the basis of the existing and emerging standards. As you move higher in the architecture towards the specification of the application then the problems of interoperability are less relevant since it is not generally necessary to have compatibility between the applications themselves. The biggest hole in the current standards work is the lack of agreement in the security domain which one might argue is fundamental to the application platform. We will discuss this area however in more detail in a subsequent part of this series.

The physical characteristics of an IC card are defined in ISO 7816 part 1. This standard applies to the ID - 1 identification card specified in ISO 7810 and includes cards which may have embossing or magnetic stripes. Whilst we are all familiar with the use of imprinters to obtain a printed version of the embossed characters on some paper voucher, their viability on an IC card must be questionable. The IC module in a smart card is like any other electronic component and is not normally expected to be hit with a hammer at regular intervals. Even the embossing process itself is mechanically stressful and must raise serious doubts over the appropriate migration strategy.

The physical properties of the contact IC card are referenced against earlier card standards and we will look at each of them in turn.

ISO 7810 Identification cards - Physical characteristics (1985)

This standard specifies the physical characteristics of identification cards including card material, construction, characteristics and nominal dimensions for three sizes of cards (ID -1, ID -2 and ID -3). It is the ID -1 card that forms the basis of ISO 7816 -1.

The principal parameters of ISO 7810 are the dimensions of the ID -1 card which are defined to be, 85.6mm x 53.98mm x 0.76mm

ISO 7811 Identification cards - recording techniques (1985)

This standard is in five parts and covers the specification of the magnetic stripe and the card embossing.

Part 1 Embossing

This part specifies the requirements for embossed characters on identification cards for the transfer of data by imprinters or by visual or machine reading.

Part 2 Magnetic stripe

This part specifies characteristics for a magnetic stripe, the encoding technique and coded character sets which are intended for machine reading.

Part 3 Location of embossed characters on ID -1 cards.

As the title implies this part of the standard specifies the location of embossed characters on an ID -1 card for which two areas are assigned. Area 1 is for the number identifying both the card issuer and the card holder. Area 2 is provided for the cardholder identification data such as his name and address.

Part 4 Location of magnetic read only tracks - tracks 1 and 2

This standard specifies the location of the magnetic material, the location of the encoded data tracks and the beginning and end of the encoding.

Part 5 Location of read - write magnetic track - track 3

This standard has the same scope as part 4 except that it defines the read - write track 3.
ISO 7812 Identification cards- numbering system and registration procedure for issuer identifers (1987)

This standard relates to the card identification number or PAN (Primary Account Number) which consists of three parts, the issuer identifer number (IIN), the individual account identifier and the check digit.

ISO 7813 Identification cards - Financial transaction cards (1987)

This standard defines the requirements for cards to be used in financial transactions. It specifies the physical characteristics, layout, recording techniques, numbering system and registration procedures. It is defined by reference to ISO 7810, ISO 7811 and ISO 7812.

In particular the standard defines more precisely the physical dimensions of the card as follows,

■	Width	85.47mm - 85.72mm
■	Height	53.92mm - 54.03mm
■	Thickness	0.76mm ± 0.08mm

The thickness of the card is particularly important for smart card readers because of the mechanical construction of the card connector mechanism.

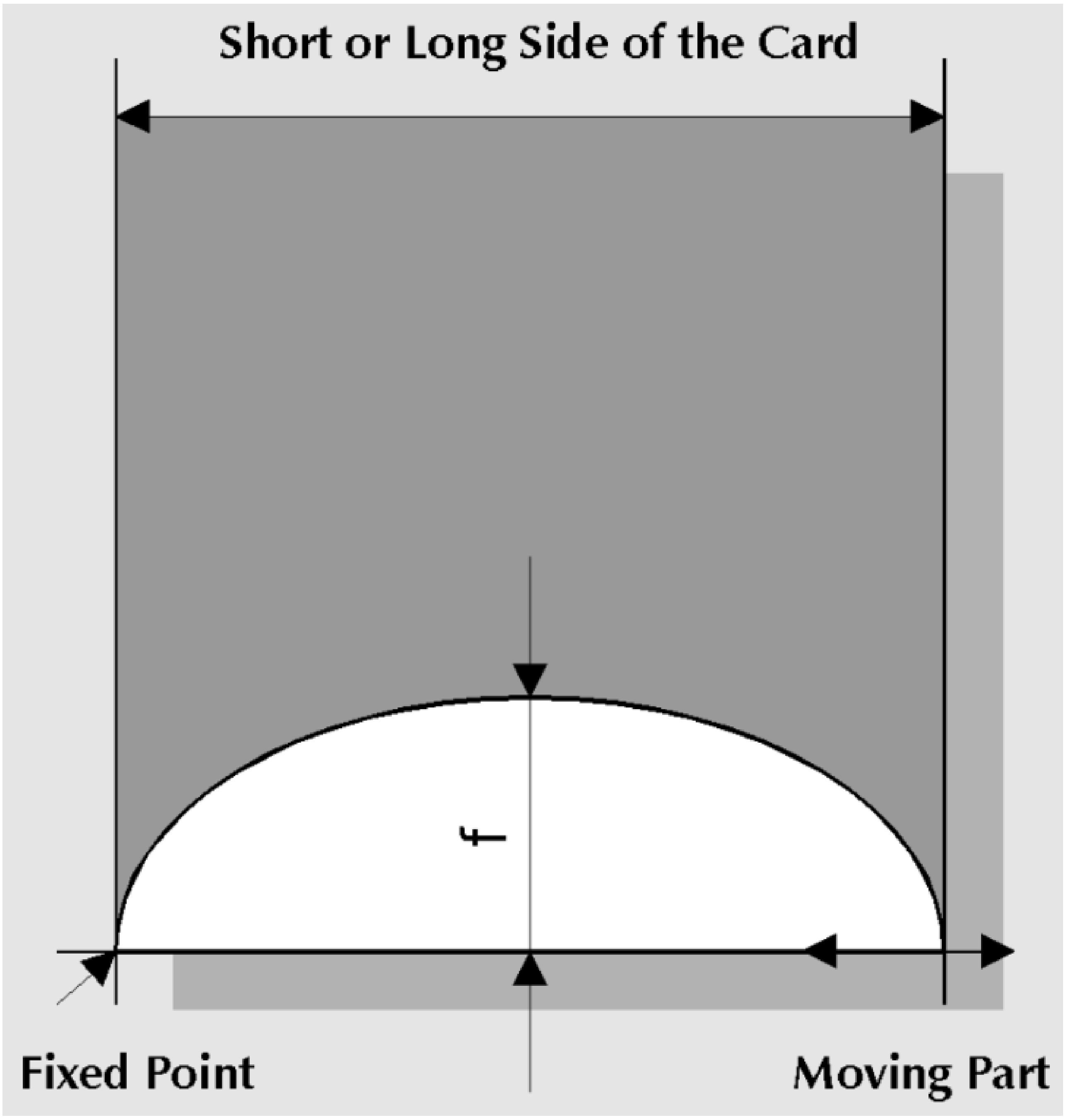


Figure 1: Bending Test

This device often consists of a movable carriage that positions the card under the connector head whilst applying the necessary wiping and pressure action. Variation in thickness or even slight warping of the card can cause communications failure.

ISO 7816 Design and use of identification cards having integrated circuits with contacts (1987)

This standard in its many parts is probably the most important specification for the lower layers of the IC card. The first 3 parts in particular are well established and allow total physical and electrical interoperability as well as defining the communication protocol between the IC card and the CAD (Card Acceptor Device).

Part 1 Physical characteristics

The physical dimensions of the IC card are defined as that specified in ISO 7813. It should be noted that the thickness dimension does not include any allowance for embossing. More particulary the slot for a card may include an extra indentation for the embossed area of the card. In effect it acts as a polarisation key and may be used to aid the correct insertion orientation of the card. This is an additional characteristic to the magnetic field sensor which operates off the magnetic stripe and is used to open a mechanical gate on devices such as ATM's where some vandal proofing techniques are required.

The part 1 standard also defines additional characteristics that should be met in the manufacturer of an IC card. These characteristics fall into the following categories:

- Ultra violet light
- X - rays
- Surface profile of contacts
- Mechanical strength (of cards and contacts)
- Electrical resistance (of contacts)
- Electromagnetic interference (between magnetic stripe and integrated circuit)
- Electromagnetic field
- Static electricity
- Heat dissipation

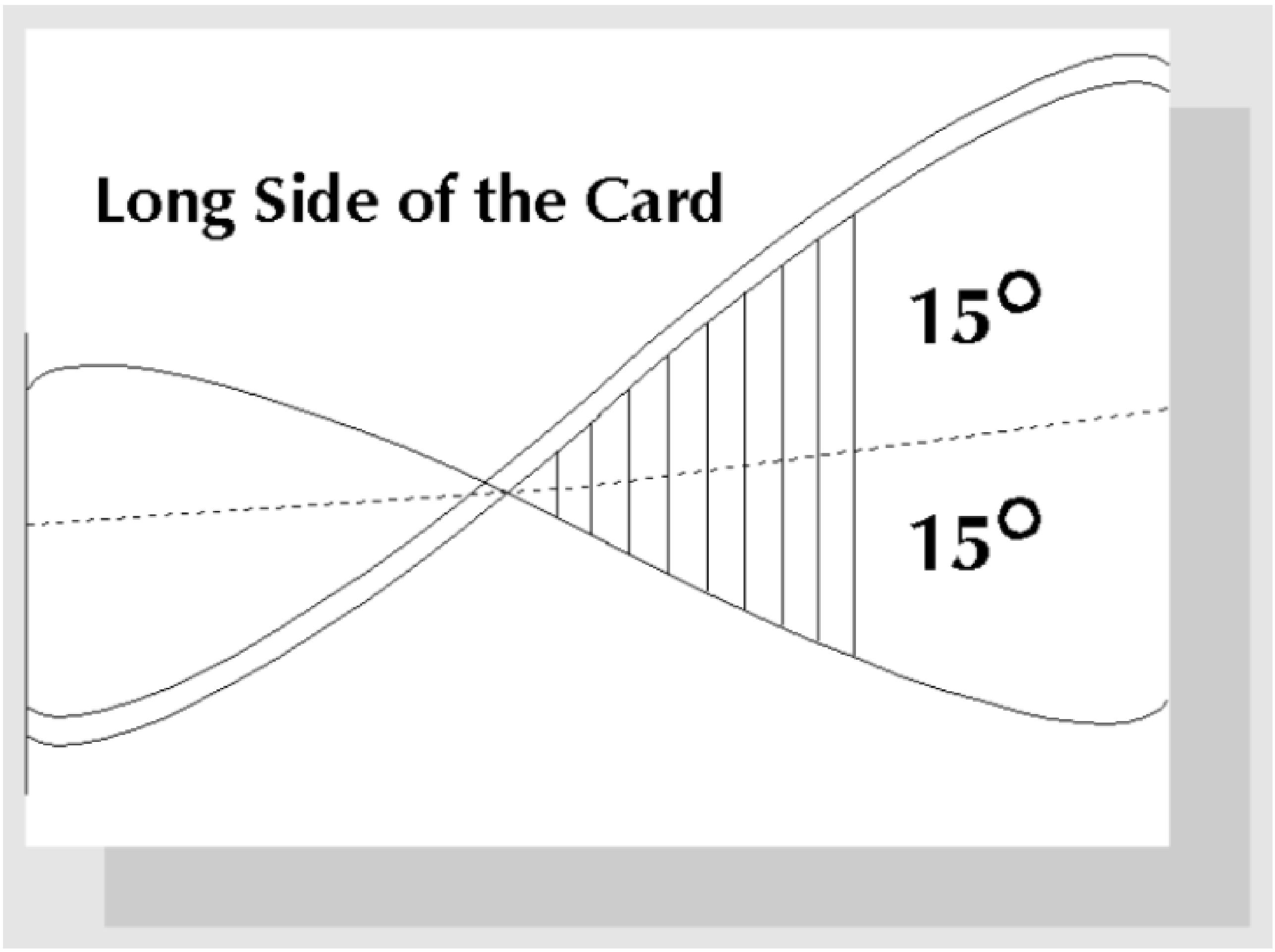


Figure 2: Torsion Test

It has to be said that this part of the standard could be improved and there is currently some work taking place in ISO on this very subject. The three most widely used tests applied by fabricators are specified in the annex to the standard,

- A1 Bending properties
- A2 Torsion properties
- A3 Static electricity

Whilst this is certainly one way of comparing cards fabricated by different companies, whether it bears any relationship to the use of IC cards in the field seems debatable.

The bending properties are tested by deflecting the card on each axis as shown in fig. 1. With a periodicity of 30 bendings per minute the card is deflected to 2 cm at its centre from the long axis and 1 cm from the short axis. The recommended test requires the card to withstand 250 bendings in each of the four possible orientations (i.e 1000 bendings in total).

The torsion properties of the card are tested by displacing the card $\pm$ 15^{o} about the long axis at a periodicity of 30 torsions per minute (fig 2). The standard requires the card to withstand 1000 torsions without chip failure or visible cracking of the card.

The resistance of the card to static electricity is defined by a test set up as shown in fig 3. The test voltage is defined to be 1.5KVolts. The specification requires this voltage to be discharged across each of the contacts in both normal and reverse polarity. The IC should still be operational at the end of the test.

One of the issues surrounding the use of the IC card relates to the temperature range for operational use. ISO 7810 defines that the ID-1 card should be structurally reliable and usable between -35 ^{o} C and +50 ^{o} C. The draft CEN standard on requirements for IC cards and terminals for telecommunications use, part 2 - application independent card requirements (EN 726-2) defines more stringent requirements for operational use as -25 ^{o} C to +65 ^{o} C with occasional peaks up to +70 ^{o} C. In addition the draft identifies multi-application cards for portable battery operated equipment to be used between -25 ^{o} C and +70 ^{o} C with occasional peaks of up to +85 ^{o} C. The word occasional is defined to mean not more than 4 hours each time and not over 100 times during the life of the card.

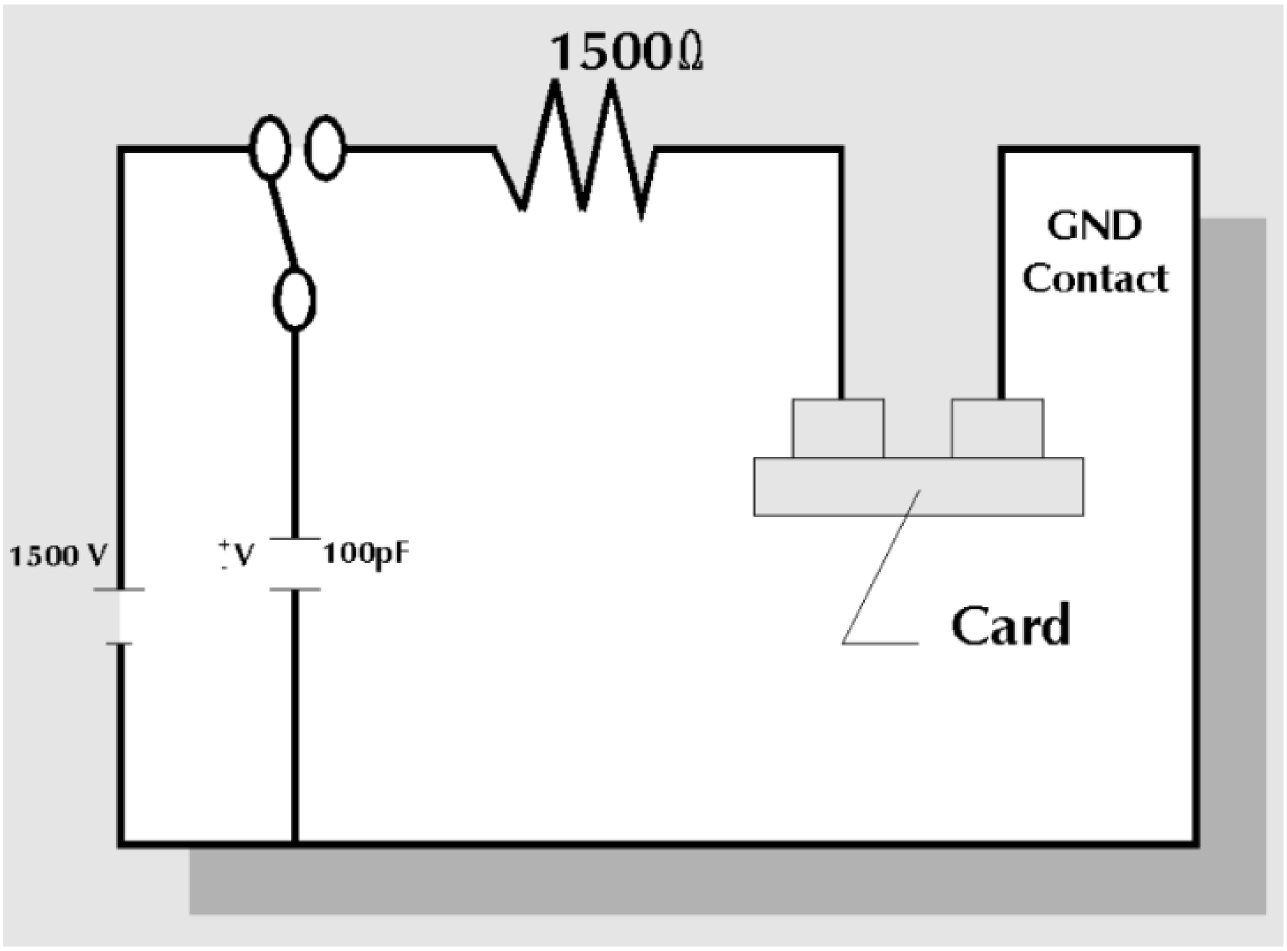


Figure 3: ESD Test

ISO 7816 Part 2 - Contact Locations and Minimum Size

This part of the standard has taken a lot of effort in order to reach agreement. Early applications of smart cards emanated in France where the Transac magnetic stripes were more central on the card than that eventually defined by ISO 7811. Unfortunately the French chip position overlaps the ISO magnetic stripe definition. As a result it was eventually agreed that after a transitional period (to the end of 1990) the position for the IC connector would be as shown in fig 4. This position is much closer to the longitudinal axis of the card. We might like to conjecture on which is the better position for the chip in terms of mechanical stress but perhaps we should just settle for agreement.

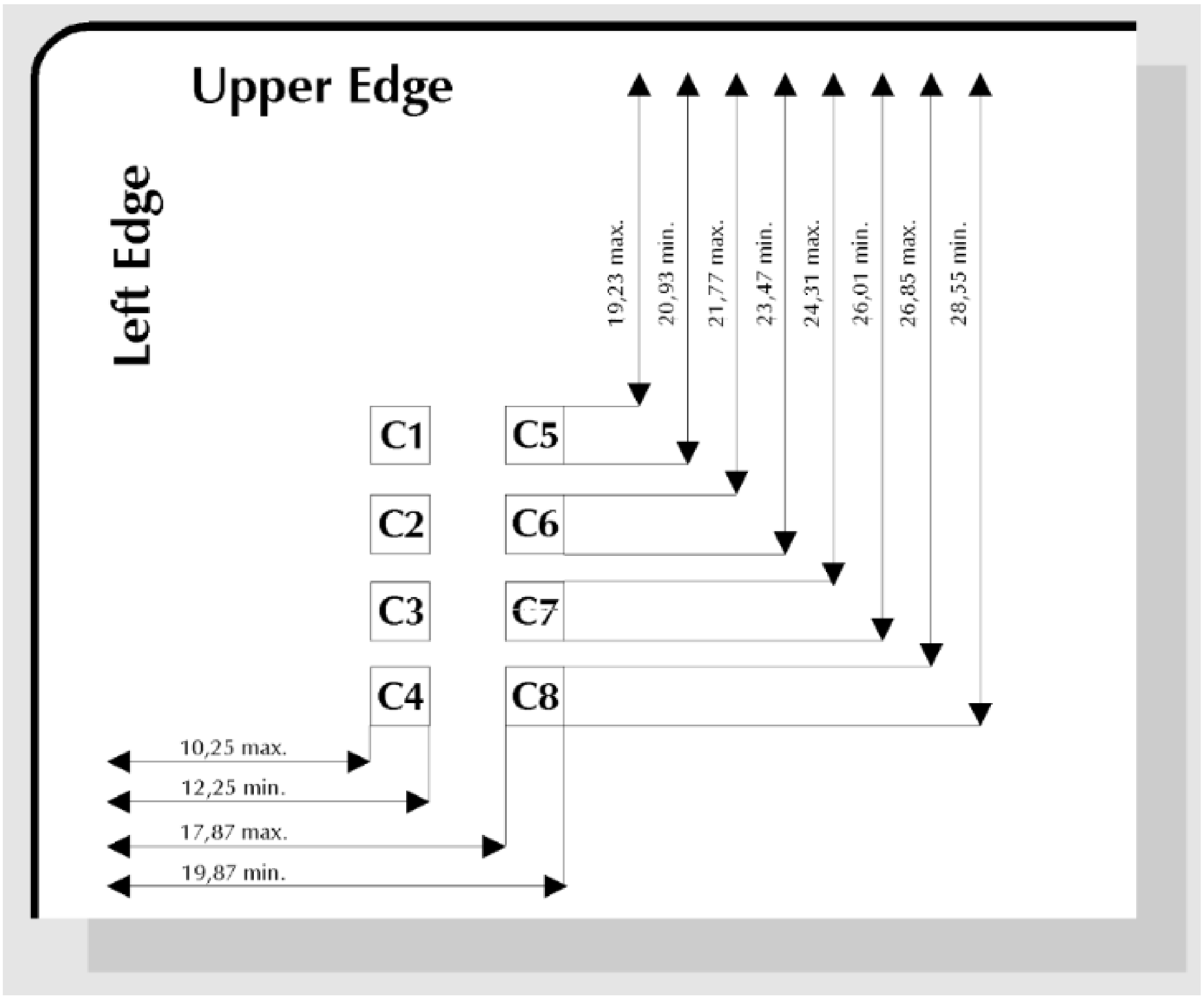


Figure 4: Contacts Location

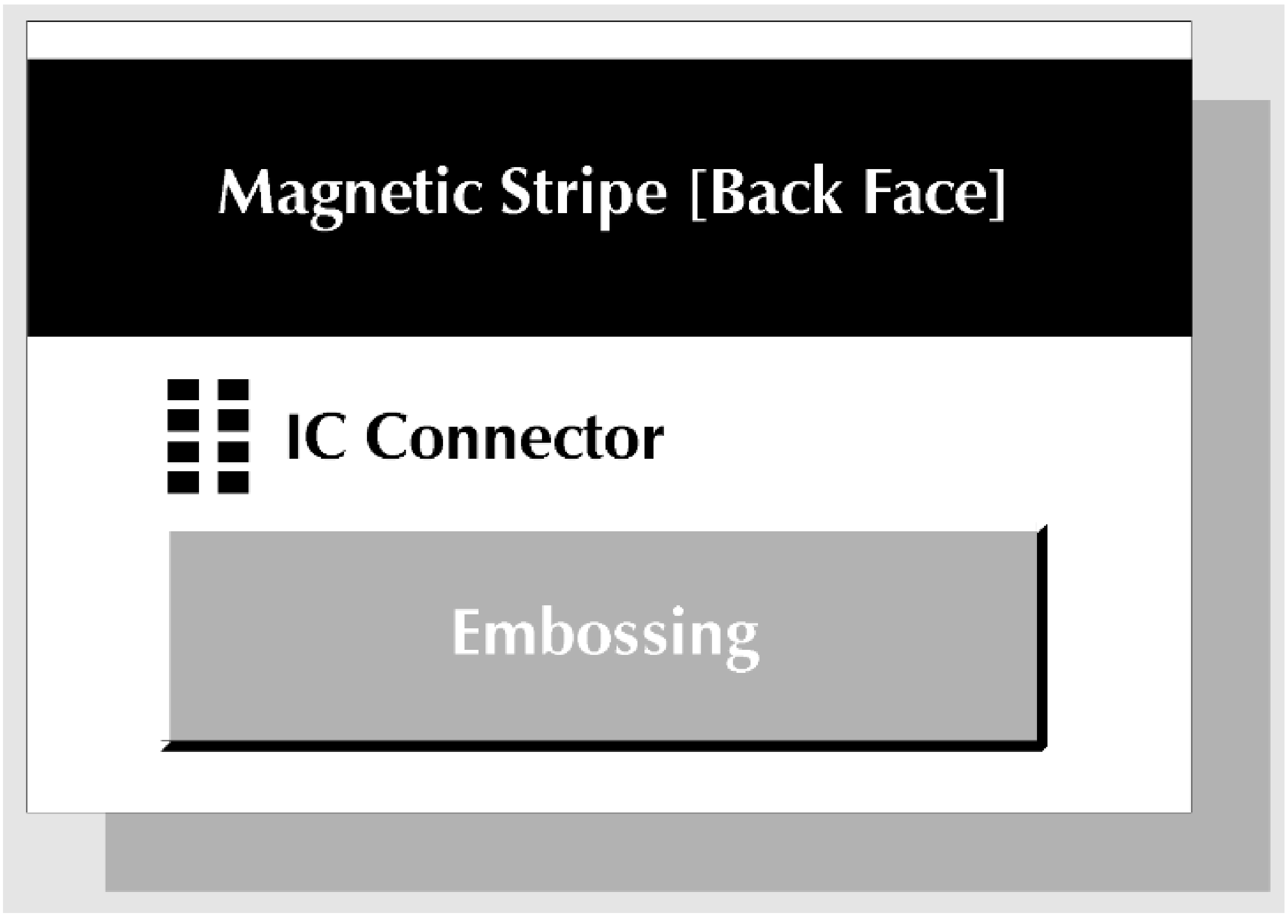


Figure 5: Relative Locations

Further problems arose in deciding on which face of the card the connector should be located. In order to avoid further delay in publishing the standard, two options were allowed to include both the front and back of the card. This anomaly has been a source of irritation and it is now widely agreed that the IC connector should be on the front of the card. For this purpose the back is defined to be the side with the magnetic stripe. The embossing is defined to be on the front of the card and therefore on the same side as the IC connector. The relative location of these components (when present) is shown in fig 5.

Smart Card Tutorial - Part 4

First Published in December 1992

Electronic Signals and Transmission Protocols.

The electronic properties and transmission characteristics of the IC card are fundamental to interoperability. These specifications are defined by ISO as part three of the 7816 standard. This standard is subject to an amendment for the T=1 transmission protocol and a proposed review for protocol type selection (PTS). The principal subjects to be considered are as follows,

- ■ Electrical characteristics
- ■ Character transmission
- ■ Answer to reset (ATR)
- ■ T=0 transmission protocol
- ■ T=1 transmission protocol
- ■ Protocol type selection (PTS)

We will consider each of these topics in turn.

IC Card Electrical Characteristics

We have previously discussed the position and definition of the IC connector and have identified 8 contacts of which 6 are currently defined,

■	V_{CC}	Power supply
■	GND	Ground or reference voltage
■	CLK	Clock
■	V_{PP}	Programming voltage
■	RST	Reset signal
■	I/O	Serial Input/Output

Power supply (VCC)

The power supply to the IC is defined to be between 4.75 volts and 5.25 volts with a maximum current consumption of 200mA. Both of these parameters have problems. Newer chip fabrication technologies are moving sub micron, 0.8um is already commercially available and 0.5um is not that far away. These chips may operate with a supply voltage of 3 volts which results in lower current consumption. Most card acceptor devices (CAD) operate at 5 volts as specified in the ISO standard. Whilst a 3 volt IC may be designed to operate between 3 volts and 5 volts, running a 5 volt IC at 3 volts is a non starter.

A current consumption of 200mA is far too high for modern electronic equipment particulary when the equipment is portable and driven by a battery power supply. Most IC cards have a power consumption of between 10mA and 20mA (at 3.58MHz). ETSI in the development of their standards have adopted a far more rigorous specification of 20mA maximum for normal use and a 10mA maximum for use in portable equipment. They further defined the concept of sleep mode (not covered by ISO 7816-3) where the IC chip can reside in a latent mode preserving volatile memory contents with a maximum power consumption of 200uA.

Clock signal

Although the integrated circuit could contain its own clock circuit for driving the internal logic, in practice most IC chips are supplied with an external clock by the interface device. It should be noted that the speed of the serial communications on the I/O line is effectively defined by the frequency of this clock. The ISO standard aligns with the use of two widely used external clock frequencies, 3.579545 MHz and 4.9152

MHz. The former frequency is the more widely used (being based on the NTSC colour sub carrier frequency) and results in a clock divider of 372 in order to produce a 9600 bit per second (not exact but within tolerance) serial communication speed. The latter frequency has a simple divisor of 512 in order to achieve a 9600 bit per second communication speed. The standard defines the situation after reset whilst allowing the frequency to be selectively changed by means of protocol type selection.

Programming voltage V_{PP}

This signal is designed to provide the high voltage required to enable writing to the non volatile memory. The more popular IC'c use EEPROM memory where the high voltage is generated by a charge pump on chip. However the EPROM memory type needs the high voltage (usually 12.5V or 21V) to be externally provided on the IC connector. There have been problems in the past with terminals supplying the wrong programming voltage with somewhat drastic effects. Because of this and the significant advantages of having a rewriteable memory the EEPROM memory is by far the most popular for IC card applications, hence the role of V_{PP} is rapidly diminishing.

The Reset Signal

The reset signal is asserted by the interface device and is used to start up the program contained in the IC ROM. The ISO standard defines three reset modes, internal reset, active low reset and synchronous high active reset. Most microprocessor ICs operate using the active low reset mode were the IC transfers control to the entry address for the program when the reset signal returns to the high voltage level. The synchronous mode of operation is more commonly met with the memory card ICs as used for telephone applications.

The sequence of operations for activating and deactivating the IC is defined in order to minimise the likelihood of damage to the IC. In particular the inadvertent corruption of the non-volatile memory (EPROM or EEPROM) must be avoided. The activation sequence for the interface device is defined as follows,

- Take RST low
- Apply V_{CC}
- Put I/O in receive mode
- Put V_{PP} in idle mode
- Apply clock
- Take RST high (active low reset)

The IC deactivation sequence for the interface device is as follows,

- Take RST low
- Take clock low
- Deactivate V_{PP}
- Put I/O in the low state
- Deactivate V_{CC}

Serial Input/Output (I/O)

The ISO standard defines a single line for the interchange of data between the IC and the interface device. This means that the line must change direction depending on whether the IC is transmitting or receiving. In practice this cannot be instantaneous and the expression 'line turnaround time' is commonly encountered in the modem world. The transmission protocol must take account of this need to turn the line around.

Character Transmission.

The transmission characteristics operated by most microprocessor IC cards are based on an asynchronous half duplex mode of operation. In the T=0 communication protocol this involves the transmission of bytes whilst the T=1 protocol defines a block mode of operation. As we have already observed the serial communication is operated by the use of a single chip connector, where the direction of data transmission has to change depending on whether the IC card or interface is transmitting data. This is referred to as half duplex communication whereas two I/O signal connectors would be required for full duplex operation where transmission can take place in both directions concurrently.

The asynchronous type of transmission is similar to that used by the serial RS232C connector met on the personal computer. Although the PC operates in full duplex mode. The transmission of a single character (defined as 8 bits) requires an overhead of several bits as follows,

- Start bit (used for character frame synchronisation)
- Parity bit (for error detection)
- Guardtime (separation between characters)

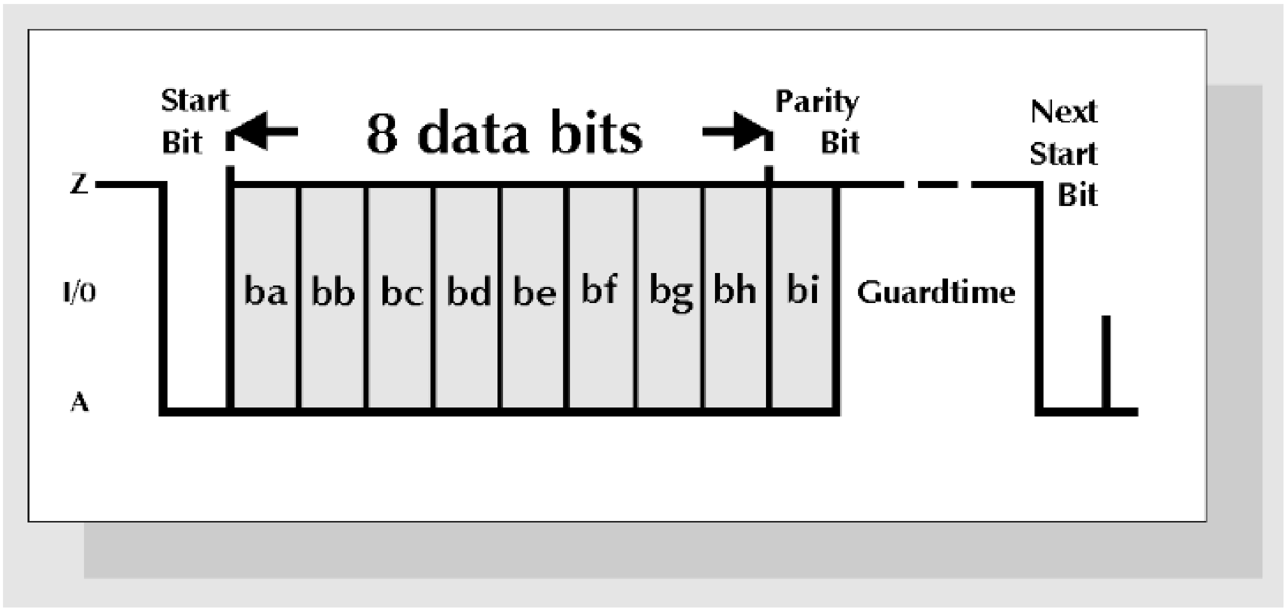


Figure 1: Asynchronous Character Frame

The format of a character frame is shown in fig.1. The receiver examines the I/O looking for the transition from the mark or high state to the space or low state. The sampling of the line is required to be such that the receiver monitors the state of the line in the centre of each bit period with a precision of + 20%. The parity bit is defined to achieve even parity which means that the number of 1's in the 8 data bits and the parity bit together results in an even number.

The guard time is defined to be equal to two bit periods (although for block mode it can be changed to a 1 bit period). This is similar to having two stop bits on a UART (Universal Asynchronous Receiver Transmitter) as used in the PC.

A more common definition of the asynchronous serial transmission at reset would be 9600 bits/second, 8 data bits, even parity, 2 stop bits with half duplex mode of operation. The half duplex refers only to data transmissions in one direction at a time which a PC is perfectly capable of managing with its UART. The RS232C interface however defines two separate wires for data transmission and reception which would need hardware modification in order to interface with the single wire IC card directly.

There is a further problem with the asynchronous character transmission that makes life difficult for a PC to act as the interface device. The 7816-3 standard defines an error detection and recovery operation (mandatory for T=0) that cannot be managed by the normal PC UART. When the receiver detects a parity error on reception it takes the I/O line to the space or low state in the middle of the first stop bit guard time. The transmitter is mandated to sample the I/O line at the start of the second stop bit guard time period. When the error condition is sensed then the transmitter should retransmit the erroneously received character. Clearly the transmitter cannot be outputting stop bits but must let the line go high during the guard time in order to sense the line state. Given the close coupling normally achieved between an IC card and the interface device one has to question whether this level of error control has sufficient benefits to outweigh the disadvantages. Error control at a higher level in the OSI model is preferable in this situation and although this could be handled at the application level the T=1 communication protocol applies error control at the frame level.

Answer to reset

After the reset signal is applied by the interface device the IC card responds with an answer to reset. For the active low reset mode the IC should respond between 400 and 40,000 clock cycles after the rising edge of the reset signal. The answer to reset is at most 33 characters (including the initial character) and consists of 5 fields,

- The initial character (TS)
- The format character (TO)
- The interface characters (TA_i,TB_ii,TC_i,TD_ii,)
- The historical characters (T1,T2. TK)
- The check character (TCK)

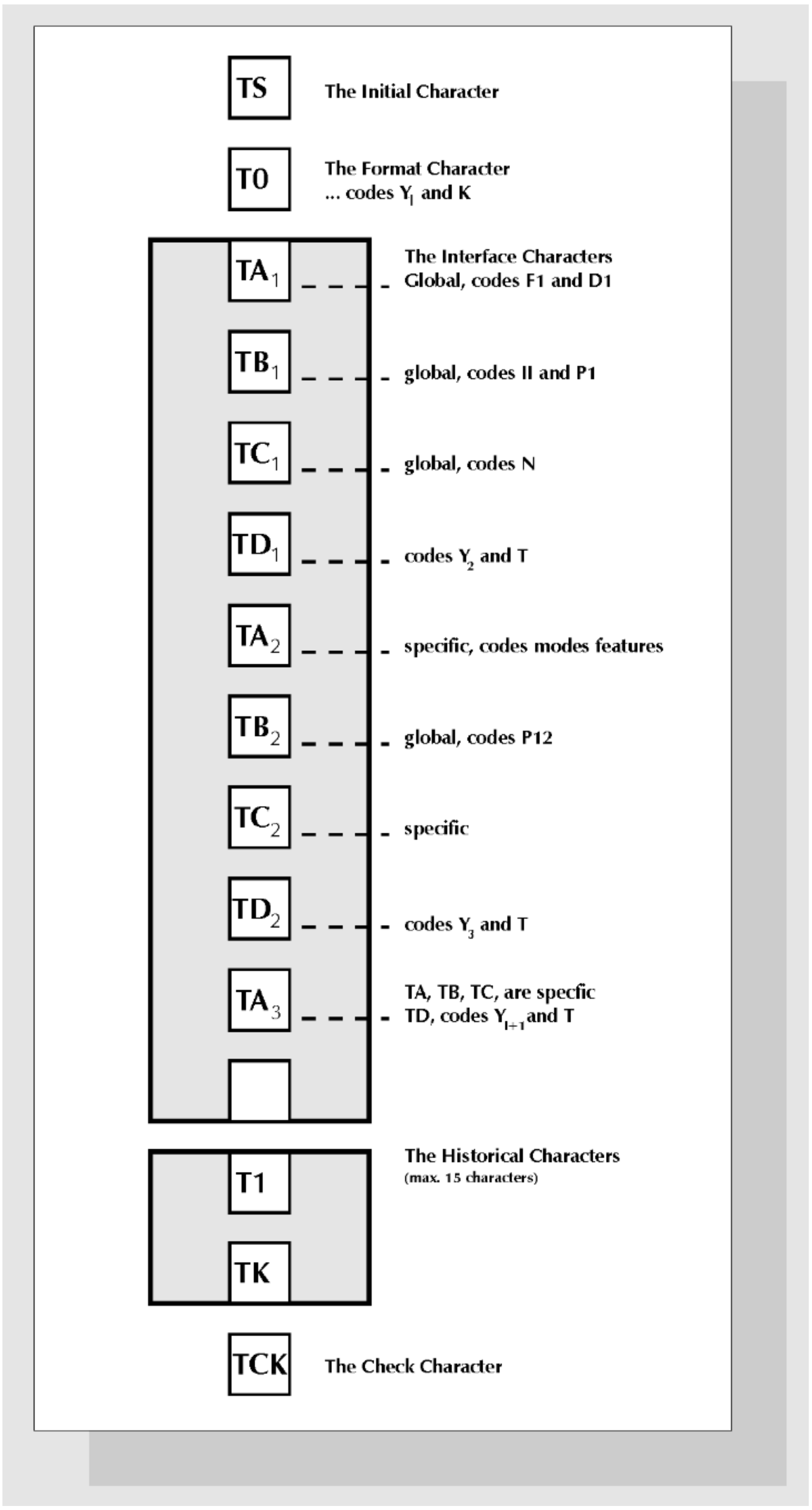


Figure 2: General Configuration of the Answer-to-Reset

Each of these fields is sent in order as shown in fig.2. The initial character TS is really a bit synchronisation pattern which may be sent in order to determine the data transmission rate (auto baud rate sensing) and also to determine the sense of the logic. The format of the TS character is shown in fig. 3. This shows the two possibilities of the direct and inverse convention. In the inverse convention where the logic level 1 is the space or low state the most significant bit is transmitted first. With the direct convention where the logic level 1 is the mark or high state then the least significant bit is transmitted first. This means that the selection of the appropriate logic sense will result in the initial character being interpreted as `3F' for the inverse convention and `3B' for the direct convention in hexadecimal coding.

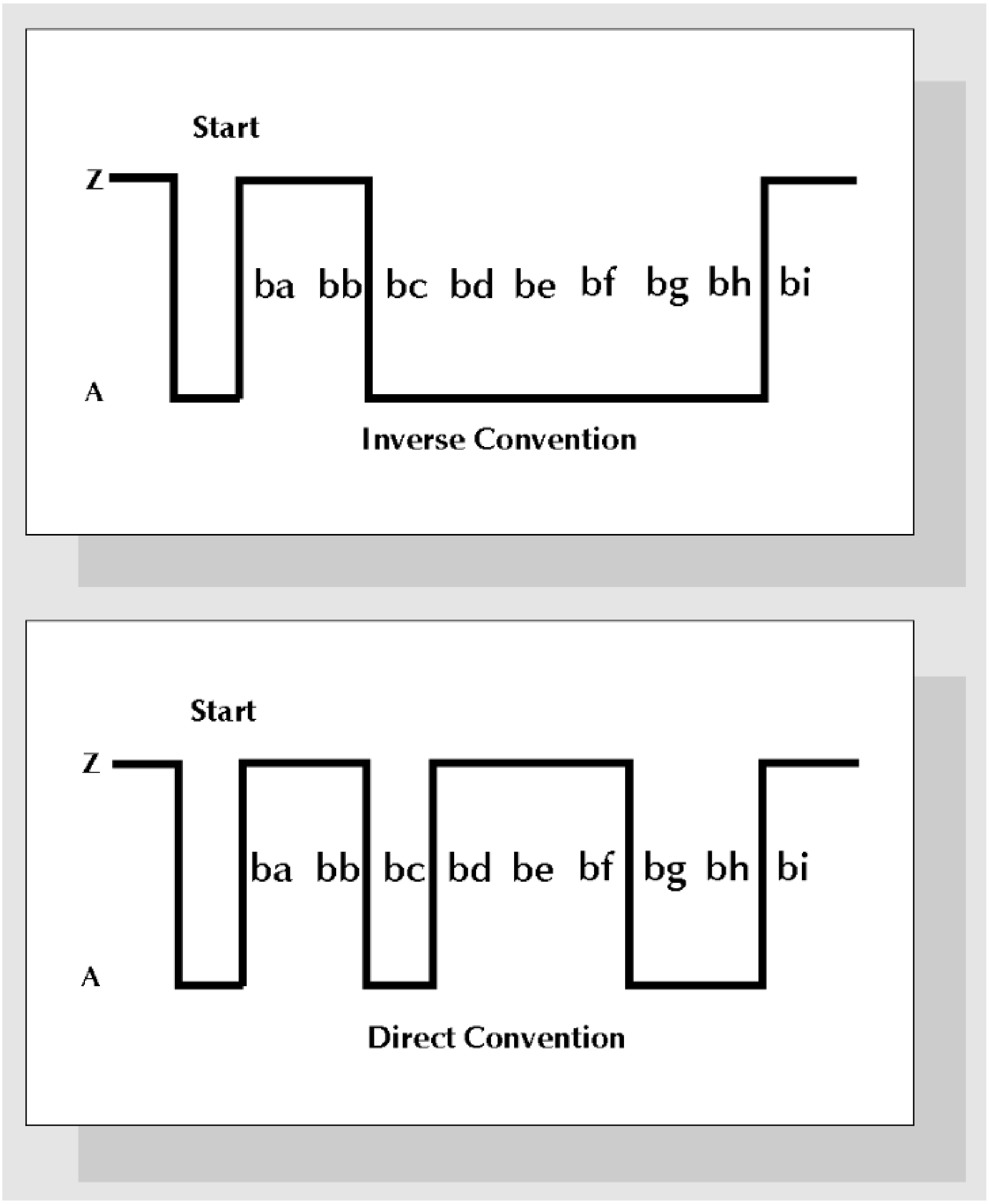


Figure 3: Initial Character TS

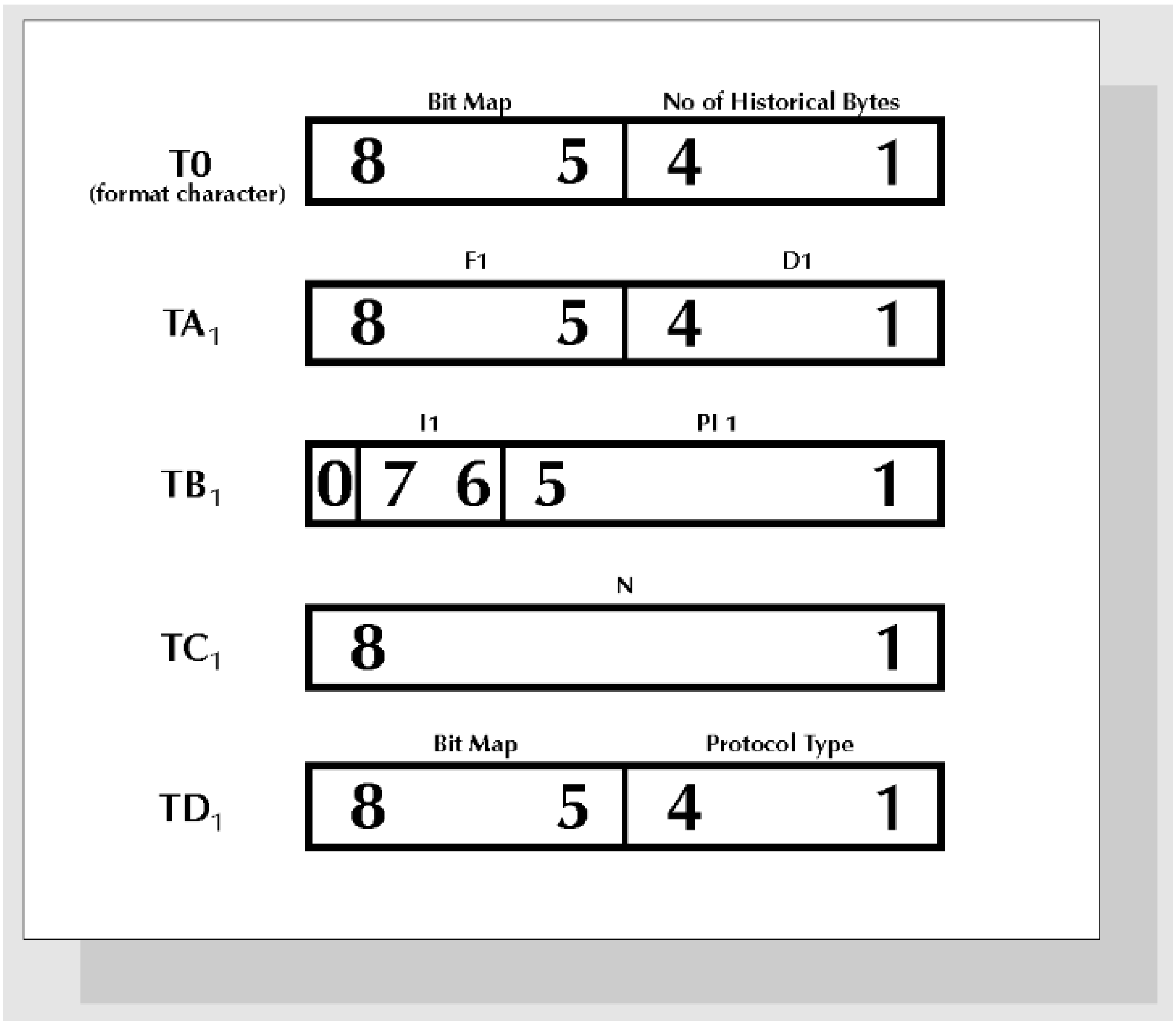
The format character TO provides information necessary to interpret the remaining answer to reset characters. The most significant 4 bits use a bit map to indicate the presence or otherwise of TA1, TB1, TC1 and TD1. For example if the most significant bit (b8) is set then TD1 is present in the interface characters field. Similarly the presence of TC1 is indicated by the state of the `b7' bit and so on.

The least significant 4 bits of the TO formal character give the number (binary encoded) of bytes in the historical field. The use of 4 bits restricts the maximum size of the historical character field to 15 bytes. The interface characters (TAi, TBi, TCi, TDi,) are the complex part of the answer to reset. They carry information relating to the available communication protocols as well as the programming voltage and current parameters for the EPROM. There is currently a proposed revision to the ISO 7816-3 to remove ambiguities and to ensure an effective method of operation for changing the protocol type and the protocol parameters. Much of the complexity is brought about by the desire to achieve backward compatibility with commercial implementations of the T=O communication protocol. At the current time there are commercial applications running either the T=O or T=1 communication protocol whilst multi-protocol operation is somewhat scarce.

The proposed revisions to the standard may alter this situation. We will discuss the interface bytes and protocol type selection against these proposed revisions but readers are warned that these recommendations are only provisional.

The interface bytes (which are optional) are defined in fig.4. The T0 and TDi characters contain bit maps which indicate the presence or otherwise of the following TA_1, TB_1, TC_1, and TD_1 bytes.

The TA_1, TB_1, TC_1, and TB_2 characters are referred to as the global interface bytes and are fundamental to the operation of the card.



TA_1 defines the basic characters of the serial transmission, FI is the clock rate conversion factor and DI is the bit rate adjustment factor. The binary encoded fields are compared against tables supplied in the standard to achieve actual values for F and D as defined below,

$$\textit{Initial etu} = \frac{372}{f} \text{ sec } (f \textit{ usually} = 3.579545MHz)$$

$$\textit{Work etu} = \frac{1}{D} \times \frac{F}{f} \text{ sec}$$

An elementary time unit (etu) is the nominal bit duration used in the character frame. Thus as described previously one character frame is equal to 12 etu (1 start etu, 8 data etu, 1 parity etu, 2 guard time etu).

The default values for F1 and D1 are 1 which is defined in the tables to give a value for F of 372 and D of 1. Hence the work and initial etu are the same. At these default values the frequency of the clock should be in the range 1MHz - 5MHz.

TB_1 is used to define the EPROM programming voltage and current. The value of II and PI1 are used against tables to obtain the value of I mA and P volts. It should be noted that TB_2 is used to define the programming voltage with higher granularity (8 bits instead of 5).

TC_1 provides the value of N which defines the extra guard time to be used between successive characters. N can be in the range 0 - 254 etu. When N is equal to 255 this indicates that the minimum guard time (2 etu for T = 0 and 1 etu for T = 1) should be used. As noted previously the T = 0 communications protocol requires the extra guard time to enable the parity error detection and signalling to be implemented.

TD_1 indicates the protocol type TDI as between 0 and 15:

■	T = 0	Asynchronous half duplex byte transmission
■	T = 1	Asynchronous half duplex block transmission
■	T = 2/3	Reserved for full duplex operation
■	T = 4	Reserved for enhanced half duplex byte transmission
■	T = 5..13	Reserved for further use (RFU)
■	T = 14	Non ISO protocols
■	T = 15	Reserved for future extension

It should be noted that Japan uses T = 14 for a National block asynchronous protocol.

The TD_1 byte also contains a bit map that indicates the presence or otherwise of TA_2, TB_2, TC_2 and TD_2.

The proposed revision defines a new use for the TA_2 interface byte which has a special role in the selection of communication protocols and parameters. We will discuss this further in the communications section.

The Historical Characters

The historical characters may be used to convey information relating to the life cycle of the card. There are clearly other possibilities and the use of these characters is still subject to agreement. This subject is being considered further as part of the emerging part 4 of the ISO 7816 standard.

The Check Character (TCK)

The check character should not be sent when only the T = 0 protocol is indicated in the answer to reset. In all other cases TCK is sent as the last character of the ATR. The check character is calculated such that the Exclusive OR of all the bytes from T0 to TCK inclusive is equal to zero.

Next Edition: 1993 Volume

We will continue with a discussion of the T = 0 and T = 1 communications protocols along with an explanation of protocol type selection (PTS)

Smart Card Tutorial

First Published in January 1993

Part 5 - Communication Protocols

At the current time there are two communication protocols that are in general use,

- T = 0 asynchronous half duplex character transmission
- T = 1 asynchronous half duplex block transmission

The T = 0 protocol is the predominant protocol in France and was the only protocol specified in ISO 7816 - 3. In 1992 ISO standardised the T = 1 protocol as amendment 1 to ISO 7816 - 3. Clearly the IC card and the interface device must operate with a common protocol. The method by which they achieve a common optimum configuration has been the subject of much discussion over the last few years. This principle is intended to be achieved by the use of protocol type selection (PTS). This is effectively a special command sent from the interface device to the ICC after the answer to reset. In order to maintain backward compatibility with existing commercial systems that may only be capable of handling the T=0 communication protocol some changes are necessary to the original ISO 7816-3 standard. A new concept is proposed which identifies the principle of two modes of operation,

- Negotiable mode
- Specific mode

An ICC that operates in a negotiable mode may have its communication protocol changed by the use of the PTS command. An ICC that operates in the specific mode cannot accept a PTS command but may be put into the negotiable mode by a further assertion of the reset command.

Although the ICC indicates to the interface device (by means of TA_2) its capability to change to the negotiable mode, an existing device in the market place may however be unaware of these changes and therefore will not be prepared to reset the card.

The operation of these mode changes are shown in fig.1. It should be noted that a multi protocol card which by definition offers the negotiable mode of operation should give priority to the T=0 communication protocol. In other words if the T=0 protocol is available it should be the default protocol offered in the answer to reset.

The TA_2 interface byte which is part of the answer to reset data (discussed in part 4) gives the necessary information to allow the appropriate choice of protocol. The coding of this byte when present is shown in fig.2. In fact the presence or otherwise of this byte is used to determine the mode of operation of the card as follows,

TA_2 present in ATR -	Specific mode
TA_2 absent in ATR-	Negotiable mode

It can be seen that bit 8 in the TA_2 byte is used to tell the interface device whether the card can change to the negotiable mode.

Protocol Type selection (PTS)

Protocol type selection is used by the interface device to change the communications protocol and/or the default values of FI and DI. The PTS command must be issued immediately after the answer to reset and only applies when the IC card is in the negotiable mode.

The interface device may choose to operate by using the first indicated protocol after the answer to reset and by using the default values of F and D. This results in an implicit selection of the protocol and the communication parameters. Should the interface device wish to effect any change to this situation then it must issue the PTS command.

The PTS request consists of an initial character PTSS (coded FF_{hex}), followed by a format character PTSO, and three optional characters PTS1, PTS2, PTS3 and PCK the check character. This is shown in fig.3. The response from the ICC follows the same format as the request.

The PTS0 format character is encoded as shown in fig.3. The bit map is used to indicate the presence or otherwise of PTS1, PTS2 and PTS3. These are encoded by bits 5, 6 and 7 respectively where a logic `1' level indicates the presence of the character. The protocol type is indicated by bits 1, 2, 3 and 4 which are binary encoded for T=0 to T=15.

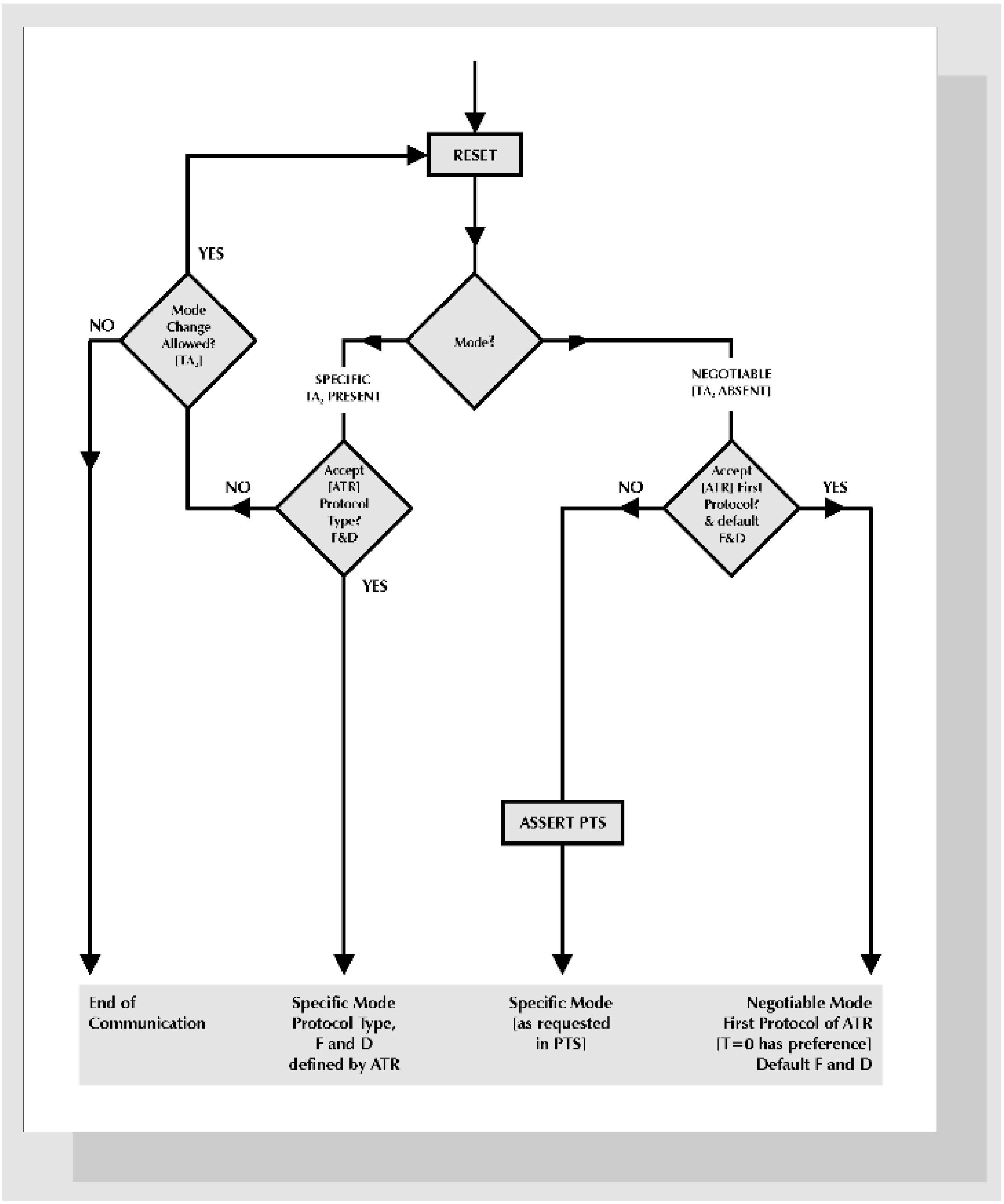

RESET
YES
NO
Mode Change Allowed? [TA1]
Mode?
SPECIFIC [TA2 PRESENT]
NEGOTIABLE [TA2 ABSENT]
NO
Accept [ATR] Protocol Type? F&D
YES
NO
Accept [ATR] First Protocol? & default F&D
YES
ASSERT PTS
End of Communication
Specific Mode Protocol Type, F and D defined by ATR
Specific Mode [as requested in PTS]
Negotiable Mode First Protocol of ATR [T=0 has preference] Default F and D

Figure 1: Modes of Operation

The PTS1 character when present is used to define the values for FI and DI as coded for TA_1 (see part 4). These parameters are used for defining the work etu (elementary time unit).

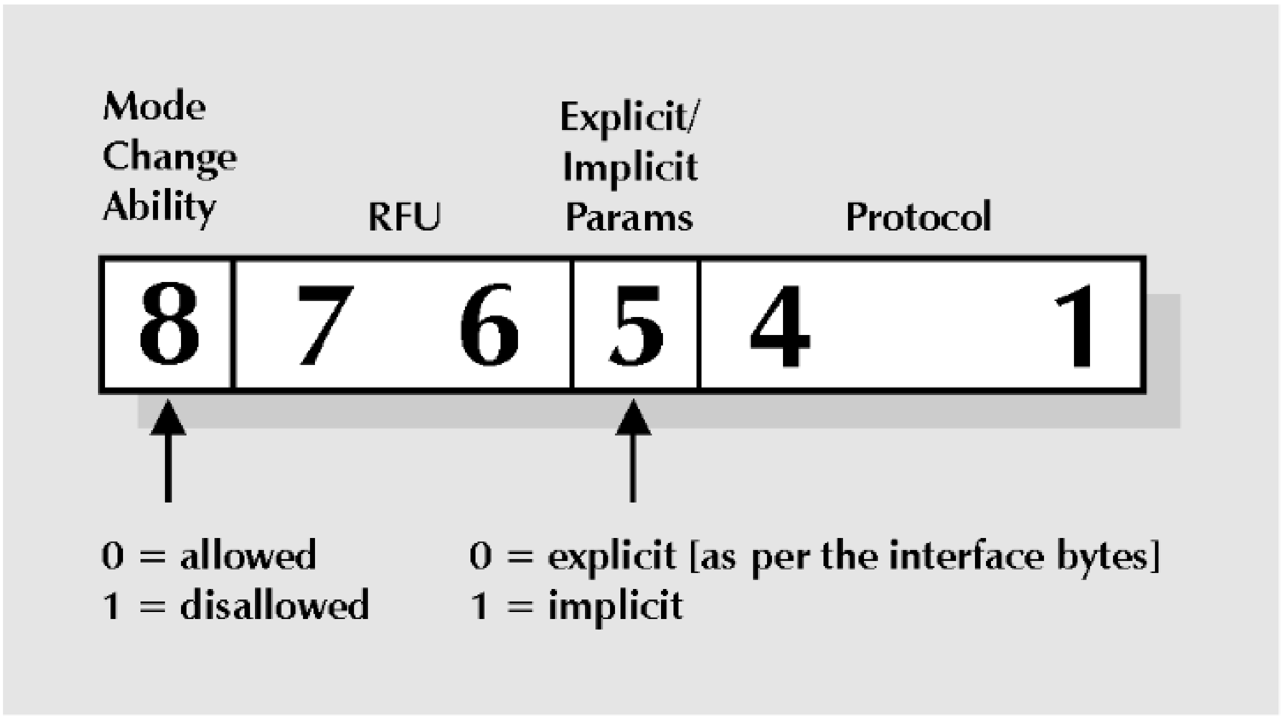


Figure 2: The TA2 Interface Byte

The check character PCK is computed such that the exclusive OR (XOR) of all the characters from PTSS to PCK inclusive is equal to zero.

When the ICC implements the PTS request message correctly it replies by echoing the same request as the response message. If bit 5 of the PTS1 response character is set to zero then the default values of F and D will be used.

The T=0 communication protocol

The interface device always initiates the command for the T=0 protocol. Interaction between the interface device and the ICC results in successive commands and responses. For this protocol, data can only flow in one direction for the command response pair. In other words, either the command message contains data for the ICC or the command request data from the ICC which is then included in the response.The direction of data flow is implicit on the definition of the command and hence both the interface device and the ICC need to have the necessary a-priori knowledge. When it is required to transfer data in both directions for a particular command then a get response command may be used after the primary command to recover the response data.

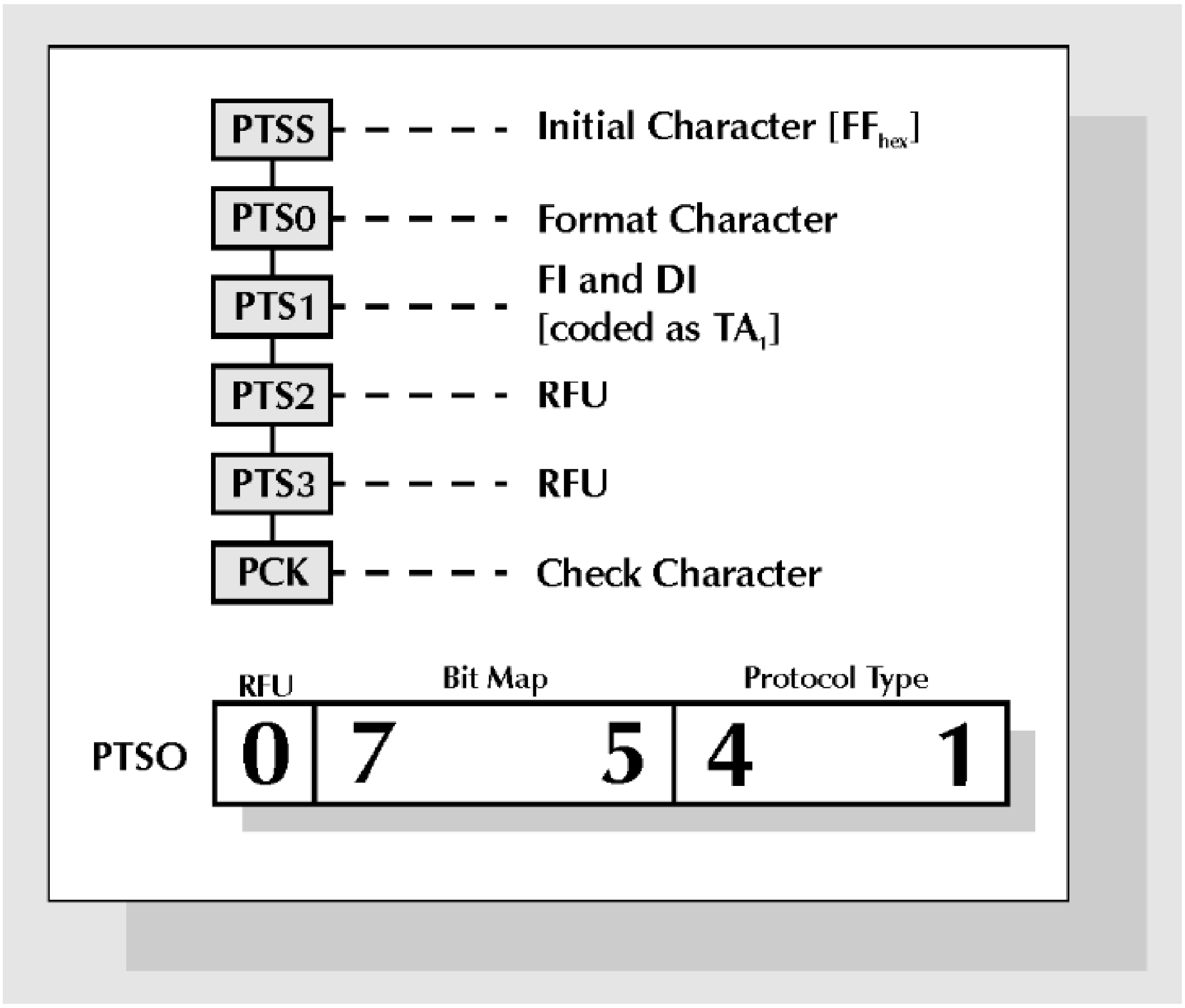


Figure 3: PTS Request and response

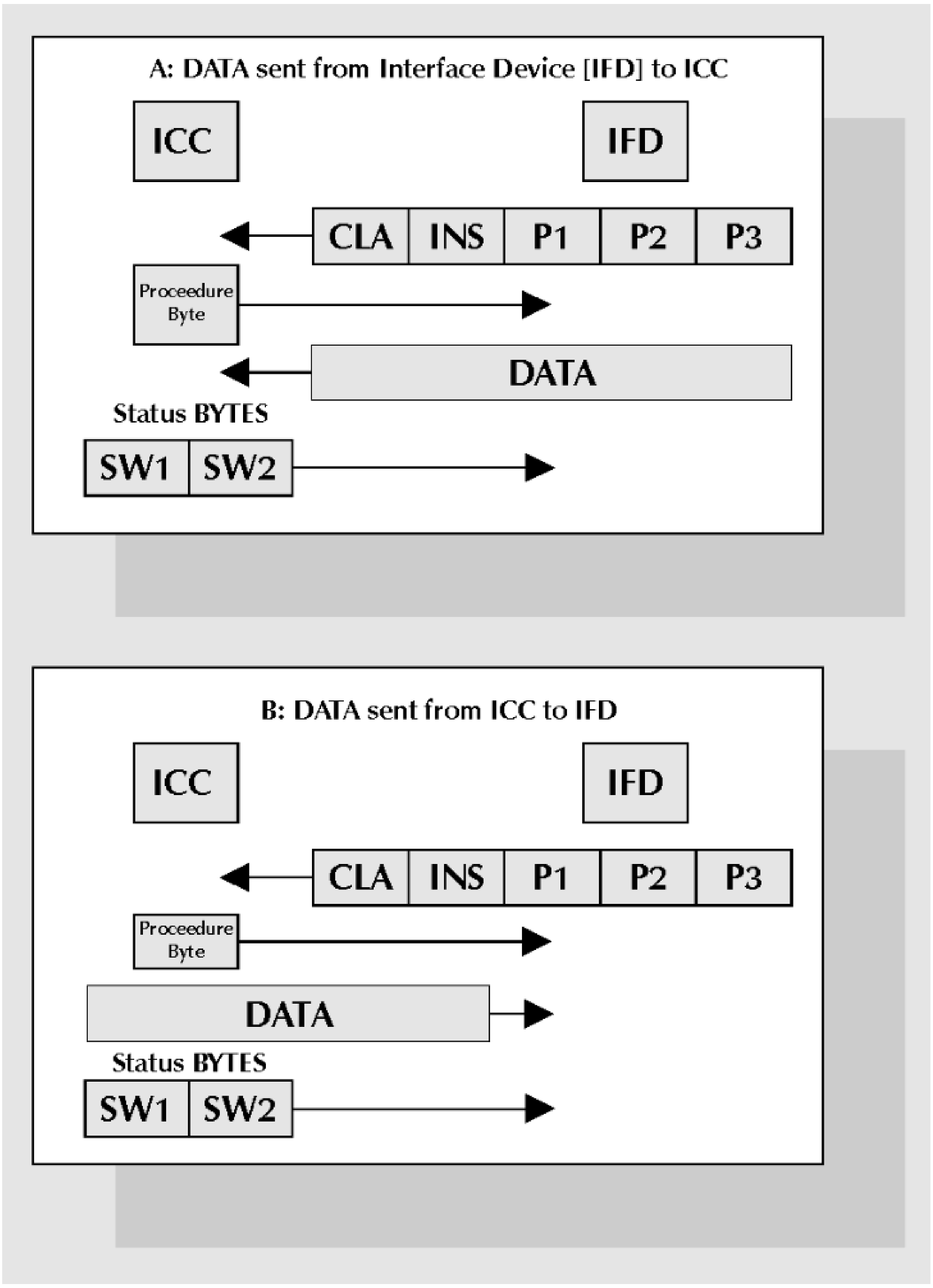


Figure 4: The T=0 Protocol

The command message consists of a 5 character header which the interface device sends to the ICC. The ICC then replies with a procedure byte after which either data is sent to the ICC, or from the ICC, depending on the particular command. This procedure byte is to allow the interface device to control the V_{pp} EPROM programming voltage. In the case of EEPROM memory this procedure byte is effectively redundant. The message flow for the T=0 protocol is shown in fig.4. The command header consists of the following 5 bytes,

■	CLA	- the instruction class (FF is reserved for PTS)
■	INS	- the instruction code (e.g read memory)
■	P1	- instruction code qualifier (e.g memory address)
■	P2	- additional INS code qualifier
■	P3	- the length of the data block

When P3 is equal to zero the data from the card will be 256 bytes. When data is to be transferred into the card then a zero data transfer is implied.

The normal condition for the ACK procedure byte is for this byte to echo the instruction byte (INS). Other options allow the interface devices to control the V_{pp} programming voltage as required. The card may optionally send a NULL procedure byte (60_{hex}) which allows further time for the processing of the command. In this situation the IFD should await a further procedure byte. The ISO standard also allows the card to send the first status byte (SW1) as the procedure byte.

There are two status bytes SW1 and SW2. These bytes are sent from the ICC to the interface device on completion of the command to indicate the current card status. The normal response is,

- ■ SW1, SW2 = 90_{hex}, 00_{hex}

When SW1 = 6X or 9X various error conditions are reported by the card. ISO 7816-3 defines 5 such error conditions,

■	SW1	= 6E	- Card does not support instruction class
	= 6D	- Invalid INS code	
■	SW1	= 6B	Incorrect reference
		= 67	Incorrect length
		= 6F	no particular diagnosis

The T=0 protocol also includes an error detection and correction mechanism. This was described in part 4 and relies on the receiver detecting a parity error upon which it takes the I/O line to the low logic level within the first etu guard time (10.5 $\underline{+}$ 0.2 etu) for a minimum of 1 etu and a maximum of 2 etu. The transmitter looks for this condition and retransmits the corrupt character.

Next month we shall complete the discussion on communications with the T=1 protocol.

Smart Card Tutorial

First Published in February 1993

Part 6 - The T = 1 Comms Protocol

The T = 1 communication is an asynchronous half duplex block transmission protocol. In terms of the OSI model this protocol operates at layer 2, the data link layer. The physical layer (layer 1) operates in the same way as for the T = 0 protocol except for the error detection and correction. In essence this protocol puts an envelope around a block of characters which allows,

- flow control
- block chaining
- error correction.

The choice of communication protocol for the ICC is still a hot topic and one has to consider what advantages can be offered by the block protocol and then to examine the price that must be paid.

The most obvious advantage of the T = 1 protocol is the ability to manage data flow in both directions. In our discussion of the T = 0 protocol it was shown that for a particular command that the data is either sent to or received from the ICC. This limitation was really due to the use of a single byte for defining the length of the data related to the command.

The T = 1 protocol also removes the T = 0 restriction of the master slave relationship where the interface device (IFD) always initiates a command to which the ICC responds. For this block protocol a command may be initiated by either the IFD or the ICC albeit within the restrictions of the protocol.

A further advantage of the T = 1 protocol is the ability to chain the blocks of data such that an arbitrarily large block of data may be transferred as the result of a single command by the transmission of the appropriate number of frames chained in sequence.

The block protocol also has a more sophisticated error management system. This allows the use of a block error detection code (EDC) and the ability to re-transmit blocks that are subject to some error condition. By comparison the T = 0 protocol has a primitive character error detection and correction scheme as described previously in the tutorial (part 4).

Clearly there is a price to be paid for this higher layer protocol. Apart from the more complex software in both the ICC and the IFD the protocol is more demanding on the RAM memory of the ICC which needs to maintain the last sent block in case retransmission is required. In general the T = 1 protocol offers advantages where the application is managing large blocks of data, particularly when it is required to pass data in both directions as part of a particular command. The efficiency of the protocol is only really apparent for larger data transmissions since the underlying physical layer is still operating in character mode as for the T = 0 protocol. The reduction of the character frame to 11 etu (elementary time units) compared with the 12 etu demanded by T = 0 has to be balanced against the administrative overhead of the frame structure which has both a prologue and epilogue.

There can be no doubt that the error control is significantly improved over the T = 0 protocol but at the lower speed of 9600 bit/second operated by many ICC's over very short transmission paths the probability of communication errors is much reduced. However it is clear that there is a move towards the use of the T = 1 protocol and it seems highly likely that this will become the predominant protocol of the future. We should not however dismiss the use of the T = 0 protocol which in some situations may well offer a more optimum technical solution. The T = 1 protocol is specified in the ISO standard ISO 7816 - 3 / AMD.1

The block frame

The frame consists of three fields,

- prologue field
- information field (optional)
- epilogue field

Prologue Field			Information Field	Epilogue Field
Node Address NAD	Protocol Control Byte PCB	Length LEN	Optional INF	Error Detection LRC or CRC EDC
1 Byte	1 Byte	1 Byte	0-254 Bytes	1/2 Bytes

The prologue field consists of three bytes,

NAD	the node address
PCB	protocol control byte
LEN	the data length

The NAD byte uses bits 3 -1 to identify the source address and bits 7 - 5 to identify the destination address. The bits 4 and 8 are used for Vpp control which will not be discussed further here. The node address byte allows the use of multiple logical channels where required otherwise both addresses should be set to zero.

The PCB byte allows the identification of three types of block frame,

An information block	(I - block)
A receive ready block	(R - block)
A supervisory block	(S - block)

The information block is the frame which is used to transmit application commands and data between the ICC and the IFD. The receive - ready block is used as an acknowledgment when the protocol is sending data as a sequence of chained blocks. The supervising block is used to establish control parameters and to effect a resynchronisation or abort status as the result of some error condition. The information block also acts as an acknowledgement byte in the non chaining mode.

The LEN byte indicates the number of bytes (if any) in the information field of the frame. Its allowed range of values are from 00 - FE hex. This allows a maximum information field of 254 bytes.

The information field is used to convey the application commands and data which we will discuss in the next part of the tutorial.

The epilogue field contains the block error detection code which may be either an LRC (longitudinal redundancy check) or a CRC (cyclic redundancy check). The LRC is 1 byte whilst the CRC occupies 2 bytes. This option is defined by the specific interface characters.

Specific Interface Characters.

In a previous part of the tutorial (part 4) we discussed the specific interface characters given by the answer to reset (ATR). The T = 1 protocol uses three of these characters to establish the necessary options before communication can take place. These bytes are assigned as follows (where i > 2),

■	TAi	=	IFSC	(default = 32)
■	TBi			
	(bit 4 - 1)	=	CWI	(default = 13)
	(bit 8 - 5)	=	BWI	(default = 4)
■	TCi			
	(bit 1 = 1)	=	CRC	option
	(bit 1 = 0)	=	LRC	option (default)

The IFSC is the information field size for the card. There is also an IFSD which is the information field size for the interface device. This has a default value of 32 bytes and can only be changed by means of an S - block request from the IFD to the ICC.

Waiting Times

The T = 1 protocol uses two waiting time parameters to help flow control,

- Character Waiting Time (CWT)
- Block Waiting Time (BWT)

The character waiting time is the maximum time between successive characters in a block whilst the block waiting time is the maximum time between the leading edge of the last character in a block sent be the IFD and the leading character of the next block sent by the card.

The character waiting time may be used to detect an error in the length of a block whilst the block waiting time may be used to detect an unresponsive card. There is also a block guard time (BGT) which is defined as the minimum time between the leading edge of the last character of one block and the leading edge of the first character in the new block to be sent in the alternative direction. The CWT and BWT are calculated from the values of CWI and BWI coded as shown previously in the specific interface bytes by means of the following equations,

- $\mathrm{CWT} = (2^{\mathrm{BWI}} + 11)$ etu

- $\mathrm{BWT} = (2^{\mathrm{BWI}} \mathrm{X}\ 960\ \mathrm{X}\ 372 / f)$ Sec + 11 etu

Where f is the clock frequency.

The minimum value for the BWT is 100 mS + 11 etu when the card operates with the default frequency of 3.58 MHz. The block guard time has a value of 22 etu such that the delay between the start of the last character of a received block and the start of a transmitted block is greater than BGT but less than BWT. Accordingly the minimum inter block time is 11 etu which is equal to one character time.

Protocol Control Byte

The protocol control byte identifies the different types of block and carries some control information including a single bit sequence number (N) and a block chaining bit (M). Other bits are used to identify transmission errors. The PCB is coded as follows.

Type	PCB (bits 8-1)								Function
I	0	N	0	0	0	0	0	0	Standard I Block
I	0	N	1	0	0	0	0	0	Chain bit set
R	1	0	0	N	0	0	0	0	No errors
R	1	0	0	N	0	0	0	1	EDC / Parity error
R	1	0	0	N	0	0	1	0	Other errors
S	1	1	0	0	0	0	0	0	Resynch request
S	1	1	1	0	0	0	0	0	Resynch response
S	1	1	0	0	0	0	0	1	IFS request
S	1	1	1	0	0	0	0	1	IFS response
S	1	1	0	0	0	0	1	0	Abort request
S	1	1	1	0	0	0	1	0	Abort response
S	1	1	0	0	0	0	1	1	WTX request
S	1	1	1	0	0	0	1	1	WTX response

The I blocks can occur as independant blocks or as part of a chain. The 'More' bit is set to indicate that further blocks are to follow. The sequence number of the sender alternates between '0' and '1' starting with '0'.

The R blocks are used to acknowledge the successful or otherwise receipt of transmitted blocks. The sequence number N carries the value of the next expected value of N where the transmitter alternates the value as mentioned above. Whilst blocks transmitted as part of a chain must be acknowledged by an R block the receipt of a successful stand alone I block may be acknowledged by an I block response. The two correspondents manage the sequence numbers of their I blocks independently alternating between '0' and '1'. The R block has three possible states as shown in the table.

The S blocks are used to invoke four control states as shown in the table. The resynch request is used by the IFD (only) to force a reset of the block transmission parameters to their initial values. A chain may be aborted by either the IFD or ICC perhaps due to some physical error such as memory corruption. The ICC may send an IFS request to invoke a change in the IFSC it can support. Similarly the IFD may send an IFS request to indicate a new IFSD it can support. The S block control also allows the ICC to request an extension to the block waiting time (BWT) that may be necessary to execute a command received in an I block. The INF field in this block contains a single byte integer value which is to be calculated as a multiple of the BWT value. In all cases the receiver of an S block should send the appropriate response block.

The T = 1 Protocol in Operation

Using the notation of the ISO 7816 standard we can show the basic operation of the protocol. A more complete definition can be obtained from the standard.

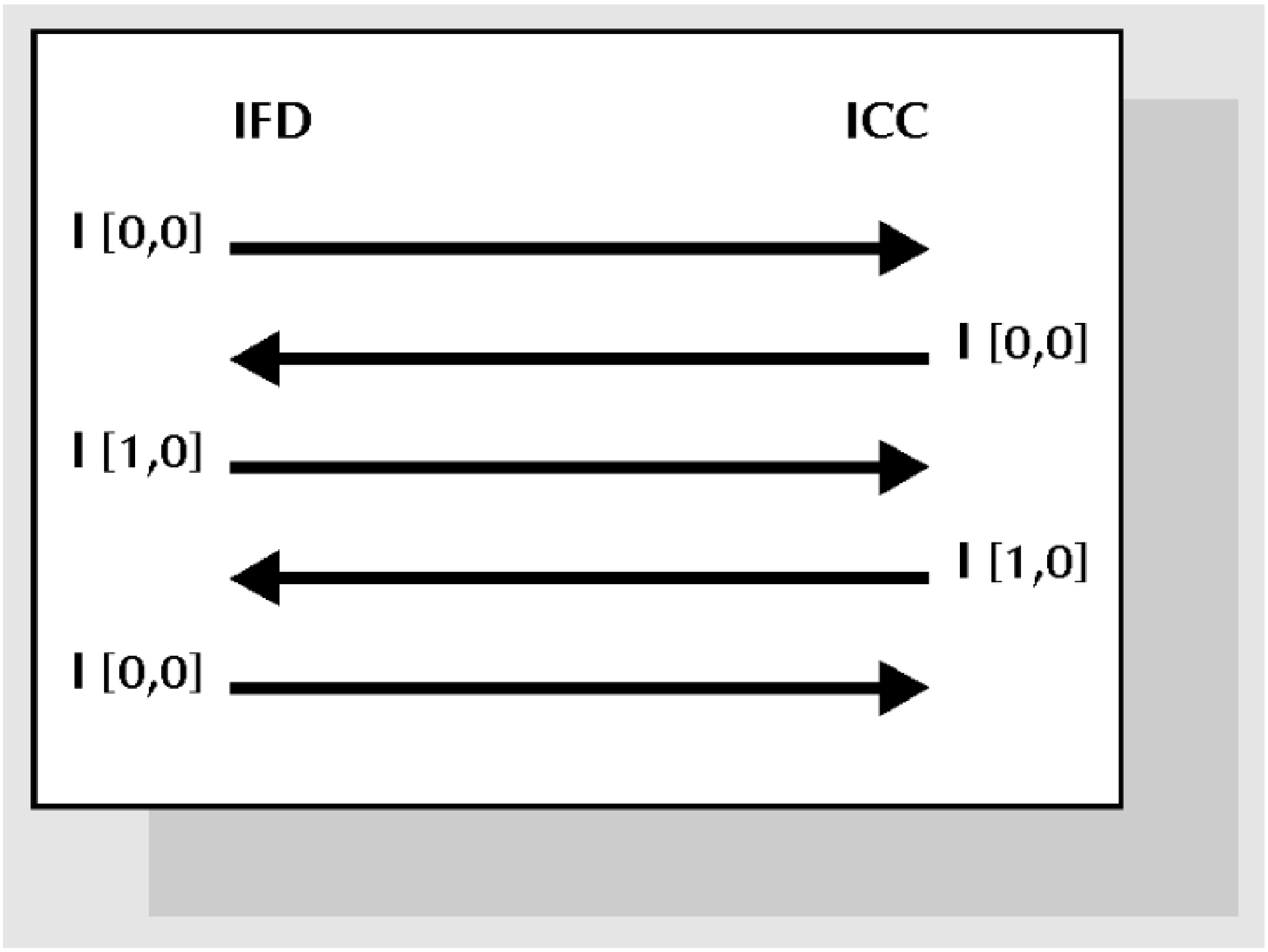
- I Blocks; I (N,M)
- Where N = Sequence number (alternately`0' and `1')

  M = More data bit

The More data bit is set when an additional I block is to follow in the chain

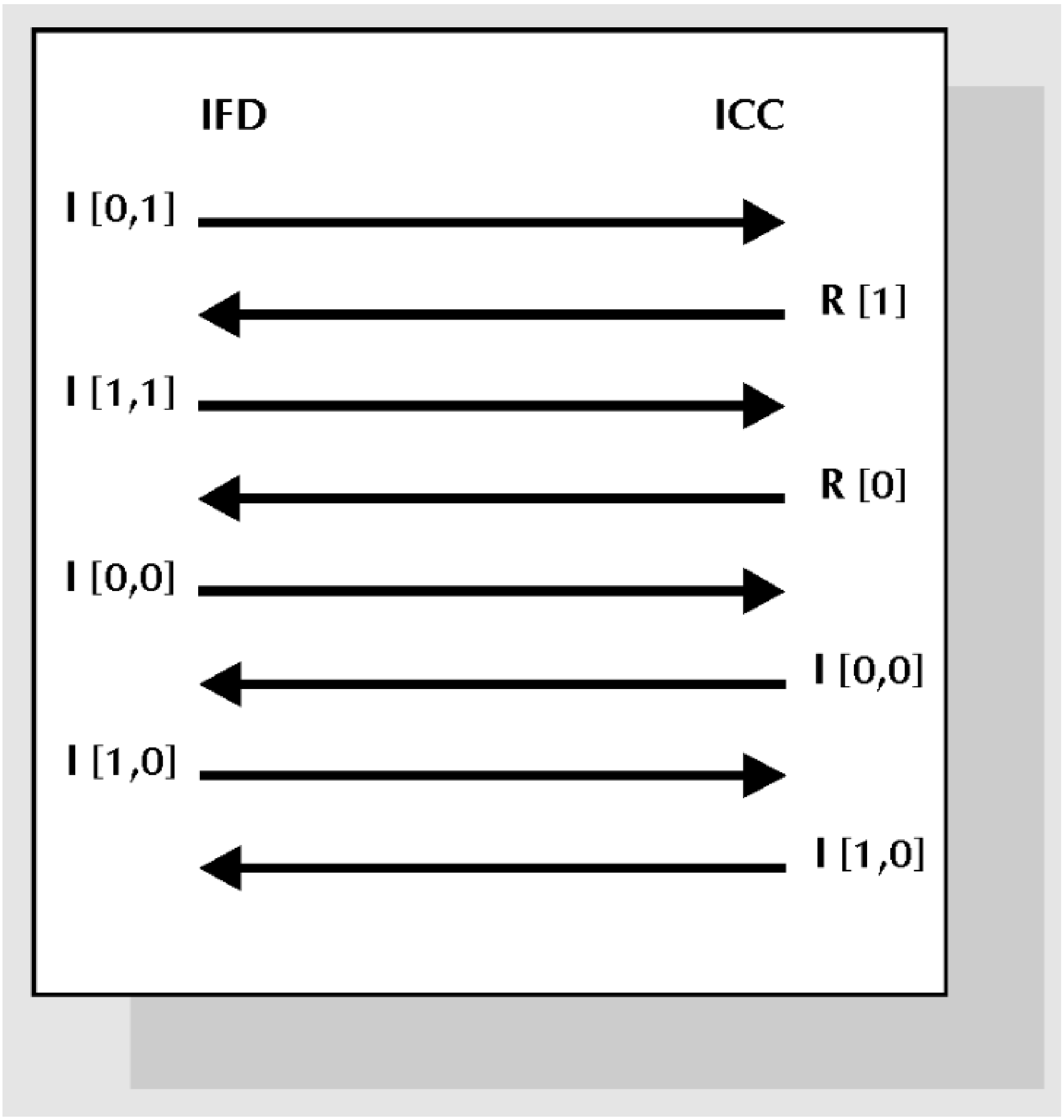
- R Block; R (N)

  Where N = Sequence number of next expected block



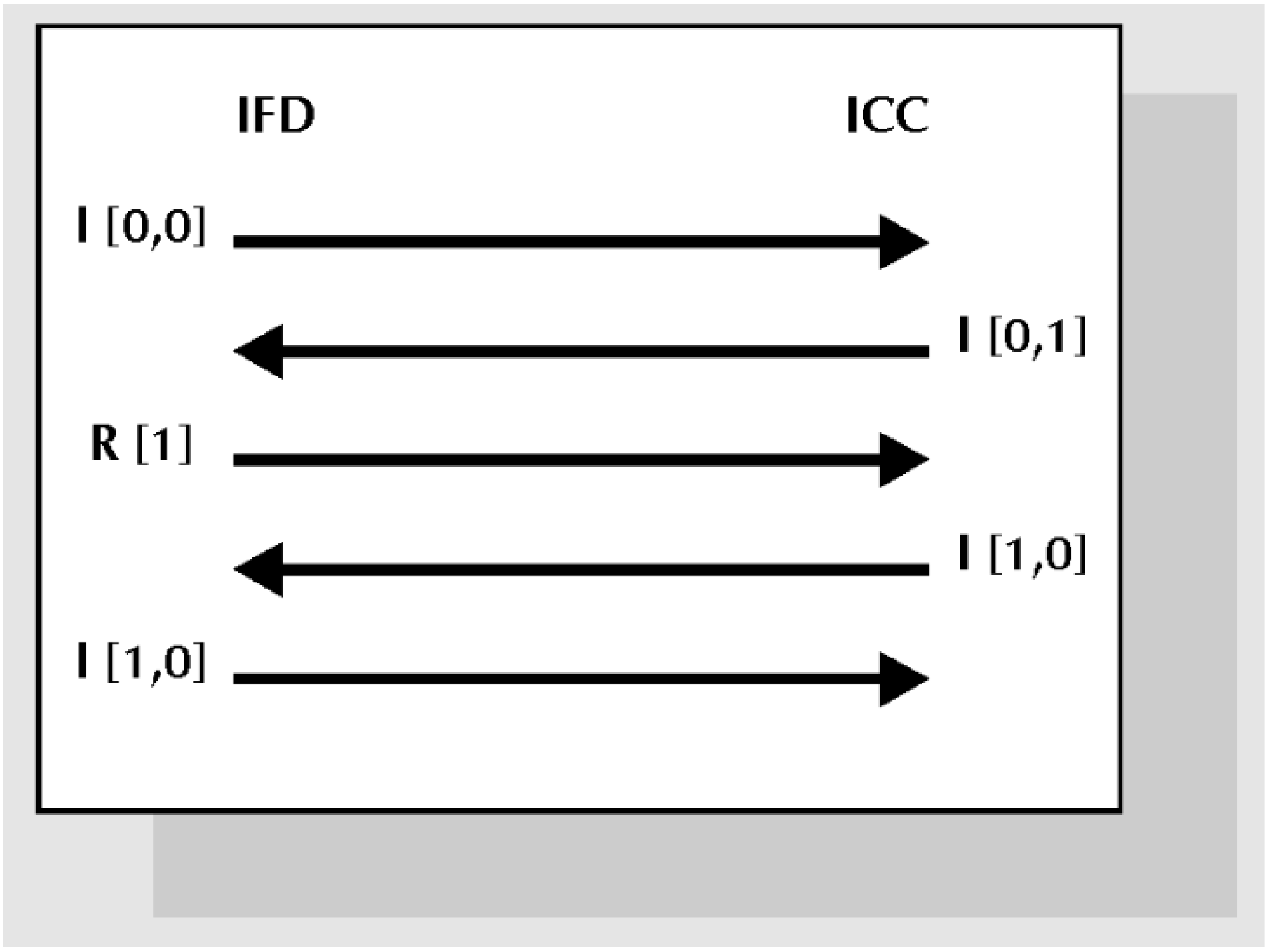
The protocol defines that the IFD and the ICC each have the right to transmit in turn where communication commences with transmission of a block by the IFD.

Normal I block transmission



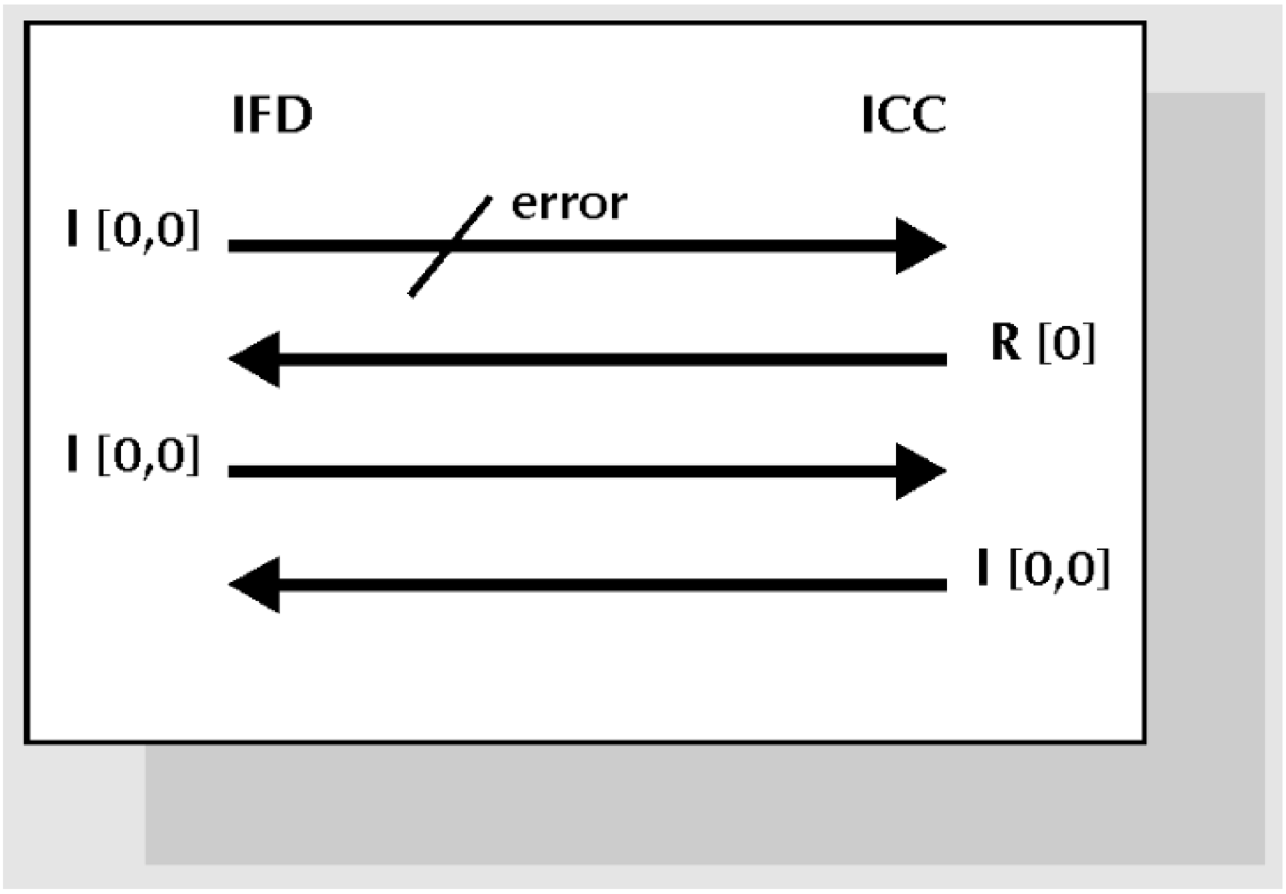
I Block Transmission With Chaining

note that an I block was used by the ICC to acknowledge the last block in the chain sent by the IFD. The ICC may send chained blocks in the same way as shown for the IFD.

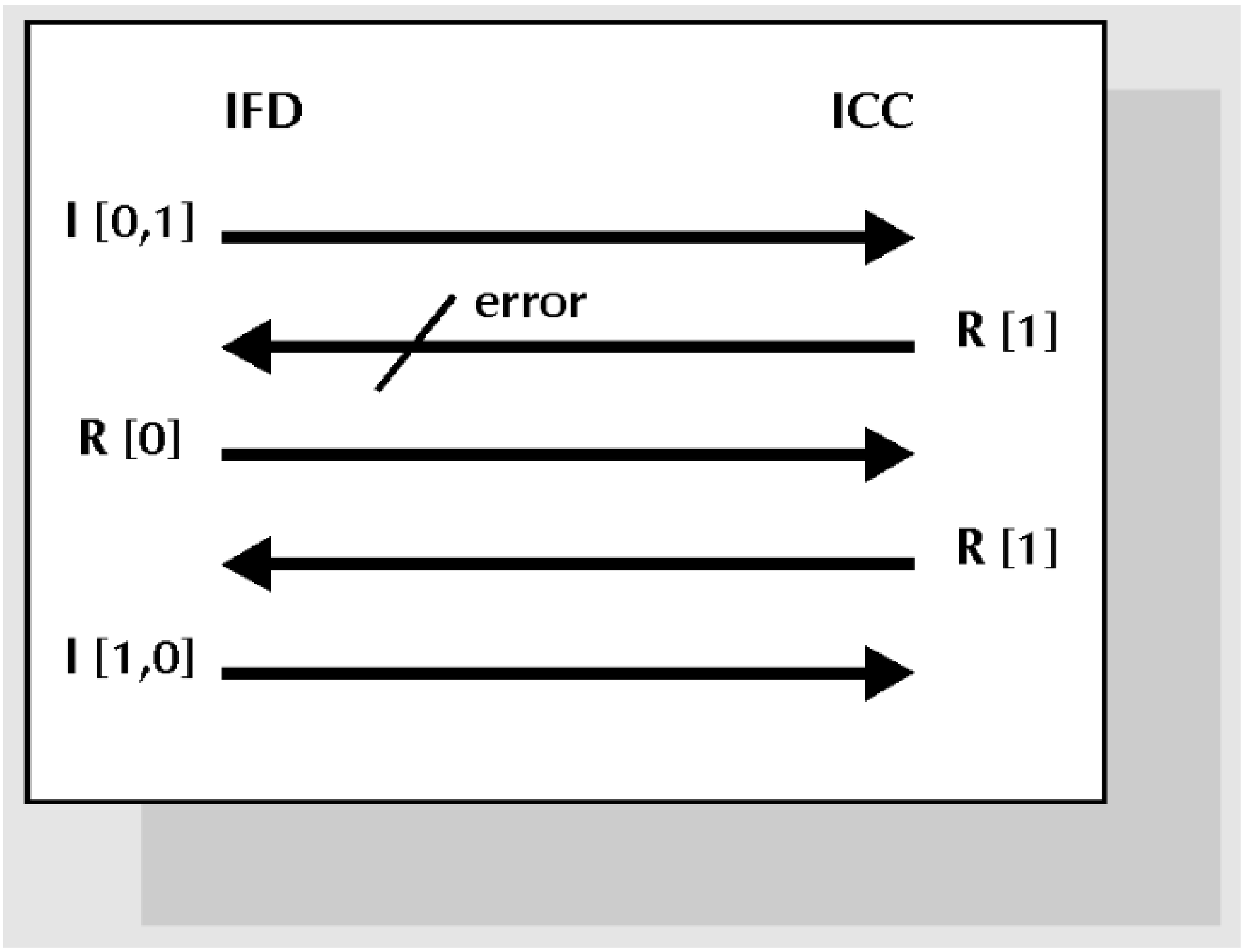


Error Handling in I Block Transmission

Error in I block receipt



Error in I block chain response



In both cases the transmitter is notified to retransmit the block received in error. There are of course a large number of possible error scenarios but they are all based on the simple concepts shown above.

Next month we will look at the proposed Inter- Industry commands.

David Everett

Smart Card Tutorial Part 7

First Published in March 1993

Inter - Industry Commands for Interchange.

So far in the tutorial we have discussed the scope of the ISO Standard 7816 parts 1,2 and 3. As we have mentioned previously any concept of interoperability requires adherence to these basic standards for the physical and electronic properties of the IC card. Whilst we encountered problems, due largely to the need to maintain conformance with early comercial implementations of the IC card system, there is none the less an overwhelming industry acceptance of these standards. We are now going to have a look at the scope of the ISO 7816-4 draft standard which is still subject to significant disagreement.

For the purpose of the tutorial we will skate around the contentious areas and concentrate on the basic principle which is really the definition of a file management system and its interaction with a user. The following discussion will examine the four basic concepts of the ISO standard,

- File structure
- Message structure
- Basic commands
- Command and data transport.

File structure

There are three categories of files,

- Master file (MF)
- Dedicated file (DF)
- Elementary file (EF)

The Master file is a mandatory file for conformance with the standard and represents the root of the file structure. It contains the file control information and allocable memory. Depending on the particular implementation it may have dedicated files and /or elementary files as descendants (See fig 1).

A dedicated file has similar properties to the master file and may also have other dedicated files and/or elementary files as descendants.

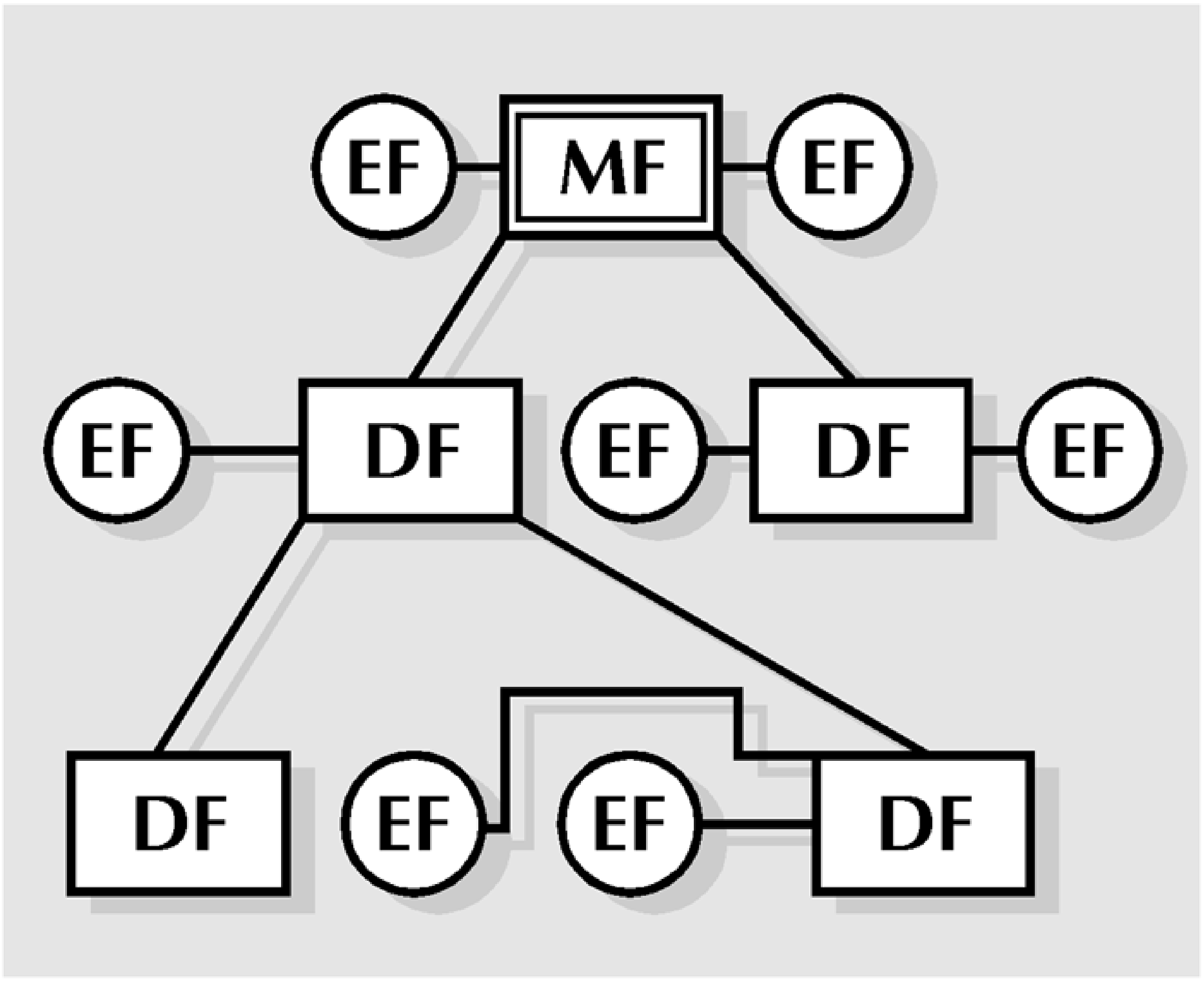


Figure 1: Logical File Organisation

An elementary file is the bottom of any chain from the root MF file and may contain data as well as file control information. An elementary file has no descendants. A number of elementary file types are defined as follows,

- Working file
- Public file
- Application control file
- Internal secret file

The working file is for storing application data whilst the public file allows data to be accessed unconditionally. The application control file always allows read access whilst the internal secret file contains data not accessible outside of the IC.

Each file is referenced by a two byte identifier which allows the path to any file to be defined from the root directory. This path concept is the same principle as used in the PC by MSDOS. Dedicated files may also be referenced by file name.

The data structure for an elementary file allows four options,

- Linear fixed
- Linear variable

- Cyclic
- Transparent

These four structures are shown symbolically in fig 2. The first three options are based on the use of records as encountered in any computer system. The transparent option just refers to a block of data without the record structure. In this case the data must be accessed by a relative address to the start of the data block. The first three structures would normally access data by reading and writing records. Where the file management system takes care of the absolute address of the data.

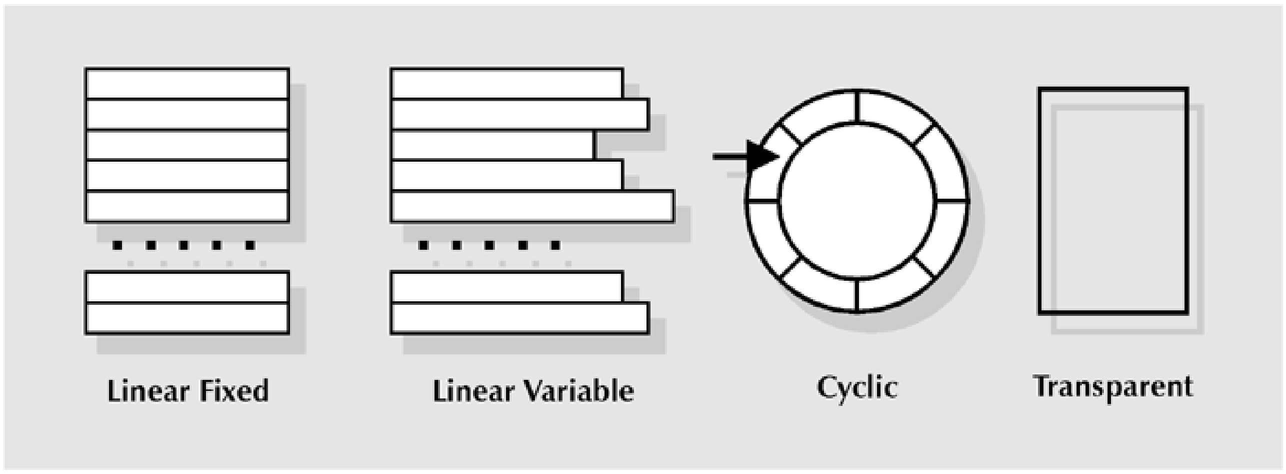


Figure 2: Elementary File Structures

This concept of a file structure really only permits the concept of reading and writing data into elementary files. The dedicated file concept allows a partition between data structures where a particular application may select a particular structure. This dedicated file may be used to control access to the data in the daughter elementary files by the use of password verification. In this sense the file structure supports the segregation of multi application data where the separate applications exist at the interface device.

This is really an incomplete picture which may support the historical use of IC cards as data carriers but does not define the principle of multi applications co-existing in the IC itself. What is really required is the concept of executing application programs in the IC and maintaining adequate security segregation between these applications. We shall return to this subject when we discuss the security of the IC card and we will show how this file structure concept may be extended in order to allow active multi - application operation.

The ISO 7816 - 4 proposed standard makes considerable use of the ASN.1 (Abstract Syntax Notation One) syntax rules for the encoding of data. These rules use the principle of TLV (Tag, Length, Value) encoding of the data field. The tag identifies this field, the Length parameter gives the size of the data (in bytes) whilst the value represents the field data. This concept allows variable length fields which may be individually identified. This is an alternative approach to a bit mapped structure where the fields and length are predefined and a single bit in a tag field is used to indicate the presence or otherwise of the field. A bit mapped approach was used in the ATR (Answer To Reset) data to indicate the presence or otherwise of the specific interface characters (see part 4 of the tutorial).

The ASN.1 encoding has a two byte overhead for each data field compared with the one bit of the bit mapped approach. Each encoding scheme has its benefits but it is clear that when data space is at a premium then the bit mapped approach is better whilst the ASN.1 encoding offers more general flexibility. Some concerns have been raised in that the use of ASN.1 may be subject to patent royalties.

The file control information referred to earlier for the MF and DF files is proposed to consist of two parts,

- The file control parameters (FCP)

■ The file management data (FMD)

The file control parameters are defined as an ASN.1 encoded data field that describes the necessary parameters such as file size, file identifier and optionally the file name. It also defines the type of file (i.e MF, DF, or EF) and the data structure (i.e Linear fixed, linear variable, cyclic or transparent). The proposed coding tables are given in the standard.

The file management data is also constructed as an ASN.1 object and may contain Inter - Industry or provider specific objects. It may be used for example to store security data for encipherment or password checking.

Message Structure

This part of the standard builds on the command response structure described in part 3 of the standard by defining the concept of an application protocol data unit (APDU). This APDU contains the command or response message and allows for all options of data transfer, as shown in table 1.

The result is an APDU which can define the length of data to be transmitted in each direction. The structure of the APDU is shown in fig 3.

The fields in the APDU are an extension of those described earlier as shown in table 2. for a command APDU.

It should be noted that this allows a number of options. The data length field may be either 1 byte (the default) or up to three bytes. This extended operation is identified by an optional field contained within the historical bytes of the ATR. Depending on the command/response data type shown in table 1. the Lc and Le field may or may not be present, for the cases 1 and 3 there is no command data. The APDU only contains those fields that are used as shown in fig 4.

The response APDU contains the response data field (if present) and the status bytes referred to in part 4 of the tutorial as shown in fig 5. These status bytes have a normal response code of 9000 hex. A number of error conditions have been identified and are described in the proposed standard.

CASE	COMMAND	RESPONSE
1	NO DATA	NO DATA
2	DATA	NO DATA
3	NO DATA	DATA
4	DATA	DATA

Table 1. Command/Response Data Option

Code	Name	Length	Description
CLA INS P1 P2	Class Instruction Parameter 1 Parameter 2	1 1 1 1	Class of Instruction Instruction code Instruction parameter 1 Instruction parameter 2
Lc field	Length of Command Data	variable ≤3	Number of bytes present in the data field
Data field	Data	variable =Lc	String of data bytes sent in the command
Le field	Length of Response Data	variable ≤3	Maximum number of data bytes expected in response

Table 2. Fields in the application protocol data unit

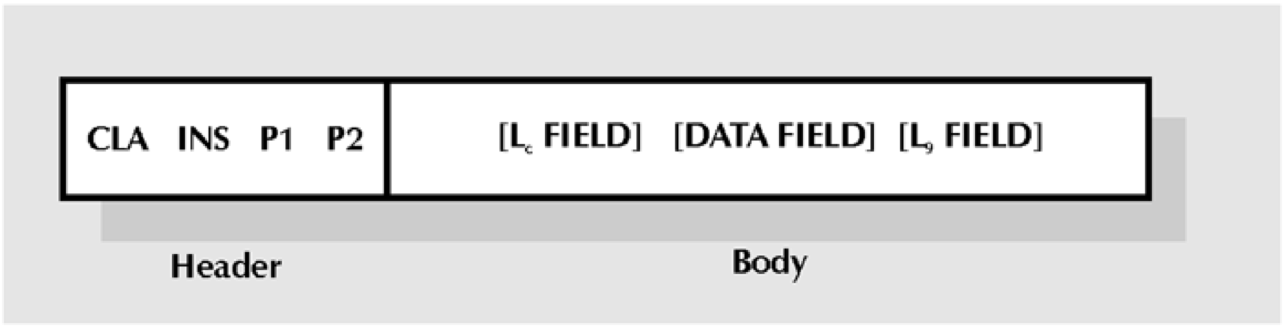
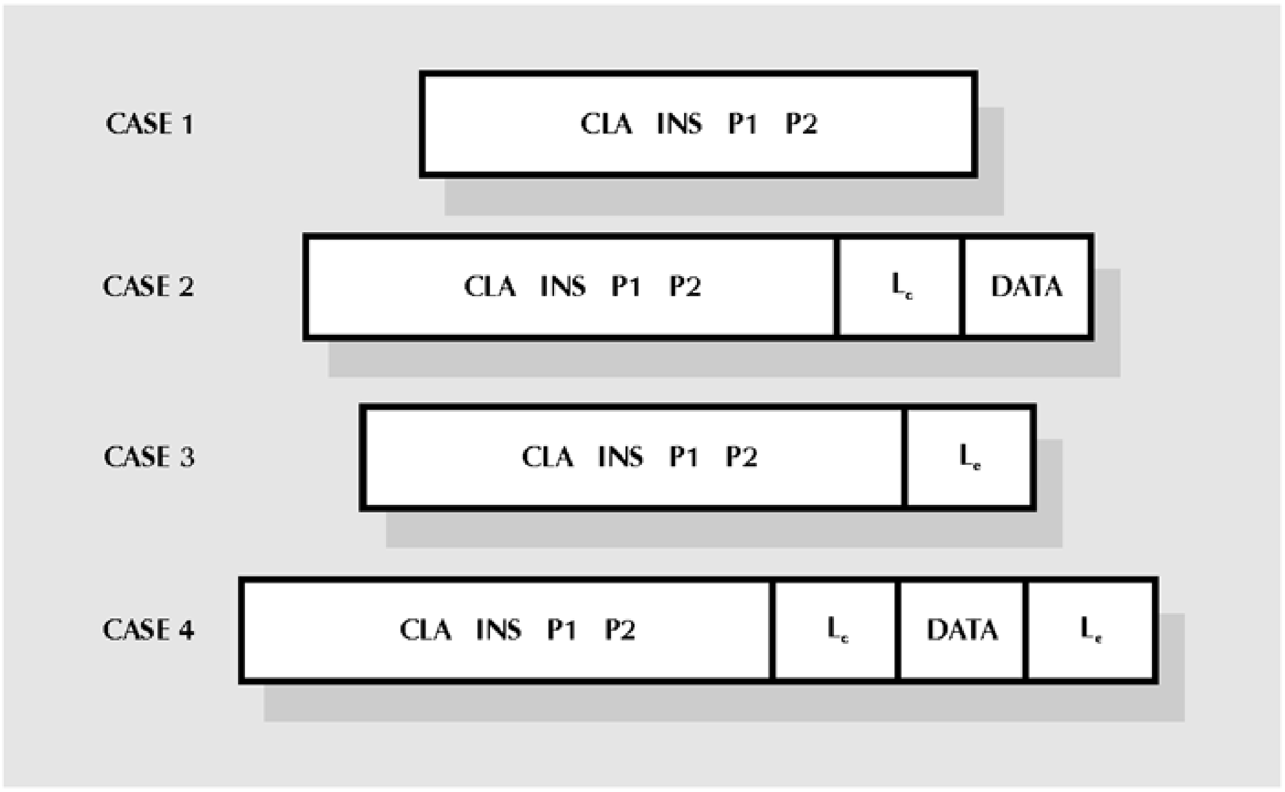


Figure 3: Command APDU



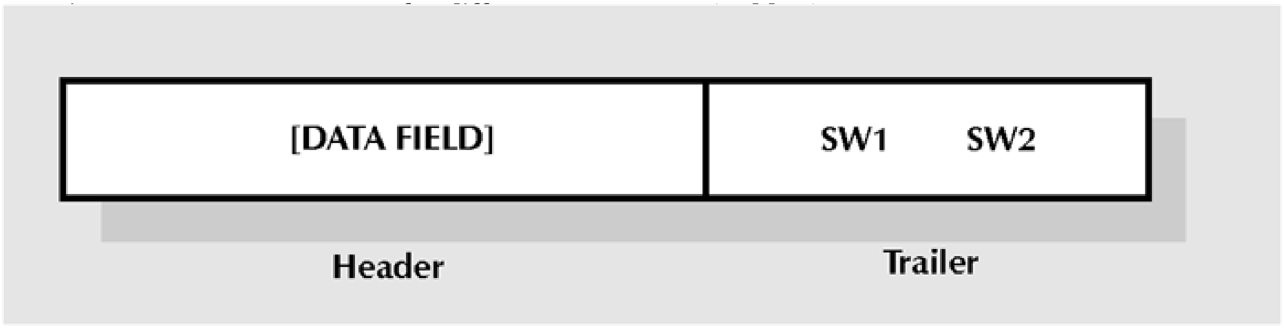


Figure 5: Response APDU Structure

Next month we will continue our analysis of the proposed Inter-Industry commands

David Everett

Smart Card Tutorial - Part 8

First Published in April 1993

Inter Industry Communications for interchange - continued.

In this part of the tutorial we will look at the basic commands described in the draft ISO 7816-4 standard. As we have mentioned previously these commands really operate against the assumption of a passive file management architecture. In other words the application in the card is really a file management system with some attention paid to access control. This of course was the only situation possible with a memory type smart card. The advent of microprocessor chips in the smart card opens up new avenues for active applications within the card. Under these circumstances it seems unlikely that such applications would allow many of the commands described here, who for example would allow any terminal to write, uncontrollably to the memory of an electronic purse application. One of the main advantages of a smart card is as a secure data carrier and in our next part we will take an initial look at security to see how everything needs to fit together.

In part 7 we described the command APDU (Application Protocol Data Unit) as shown in fig 1.

The body of this command APDU contains the data (if present) and one or two bytes defining the length of the data sent or received (see part 7 for further detail).

Observant readers will have noticed that earlier in the series when we described the T=0 communication protocol (part 5), we refer only to a fixed 5 byte header. The command APDU shown in fig.1 has only 4 bytes. Well this new part to the standard is aimed at a more general purpose application protocol data unit which allows for data to be sent in both directions (not available when using T = 0). But this part of the standard is none the less upward compatible since the 5th byte was used to indicate the length of the data which now exists in the body of the command as the 1st byte of the body of the APDU.

CLA INS P_1 P_2	Body
Header	

Fig 1. Command APDU Structure

b8	b7	b6	b5	Meaning
0	0	0	X	Message Structure & Coding as per Standard
0	1	0	X	Message Structure & Coding as per Standard Least Significant Nibble Coded as per table 3.
0	0	1	X	Message Structure Only as per Standard
0	1	1	X	Reserved for Further Use (RFU)

Table 1. Most Significant Nibble Of The Class Byte (68=0)

The Class Byte

The first byte in the header is the class byte. In the past this byte has been used by the various suppliers of IC cards operating system also as a way of identifying their particular commands.

The part 4 of the standard attempts to give more meaning to the class byte by using it to define conformance or otherwise with the structure and coding used in the standard. The following tables define the proposed use of the class byte,

b8	b7	b6	b5	Meaning
1	0	0	0	Proprietary
1	0	0	1	Proprietary
1	0	1	0	Message Structure as per Standard Least Significant Nibble Coded as per Table 3.
1	0	1	1	Message Structure as per Standard
1	1	0	0	Message Structure as per Standard
1	1	0	1	Message Structure as per Standard
1	1	1	0	Message Structure as per Standard
1	1	1	1	(Except CLA=FF_{hex}) Outside Scope of Standard
1	1	1	1	(CLA=FF_{hex}) Reserved for Protocol Type Selection (PTS)

Table 2. Most Significant Nibble Of The Class Byte (b8=1)

b4	b3	b2	b1	Meaning
X	X	-	-	Secure Messaging Format
0	0	-	-	No Indication
0	1	-	-	Secure Messaging Encoding
1	0	-	-	Proprietary
1	1	-	-	RFU
-	-	X	X	Logical Channel Number

Table 3. Least Significant Nibble Of The CLA (As requested in tables 1 & 2)

If readers cannot quickly follow the logic of these tables they are excused. However, we have recorded them here in the event that they may serve some useful purpose in the future.
Perhaps the most important thing to note is the use of CLA = FFhex which is used for protocol type selection as discussed earlier.

The Commands

The draft standard currently defines 11 commands which have varied between all combinations of mandatory and optional. They are currently all optional which says a lot about the use of this standard.

ETSI have taken a different line by defining a similar set of commands which are mandatory. The coding of the instruction byte (INS) for each of these commands is shown in table 4.

INS in hex	Meaning
OE	Erase Binary
20	Verify
82	External Authentication
88	Internal Authentication
A4	Select File
B0	Read Binary
B2	Read Record
C0	Get Response
C2	Envelope
D0	Write Binary
D2	Write Record

Table 4. Inter-Industry Commands

Erase Binary

This command is used to set part or all of an elementary file to its logically erased state. The parameters P1 and P2 in the command header APDU are used to define the offset address of the first data unit to erase. The command assumes that the elementary file (EF) has previously been selected. The data field in the body of the APDU may be used to set the offset of the first data unit not to be erased.

Verify

The principle purpose of this command is to allow the verification of a password. The password is sent as part of the command data. Here the P2 parameter is used as a code to define the whereabouts of the relevant reference data in the card. We will have more to say about aspects of security in the next part of the tutorial.

External Authentication

This command is intended to authenticate an external identity (e.g the interface device (IFD) or terminal) using the challenge response technique. The IC card sends a random number (for example) to the IFD which then encrypts the number using its secret key. The resultant cipher is returned to the IC Card (using the external authentication command) which using the same key can check its corrctness and hence the authenticity of IFD. This proves that the IFD and IC card are members of a set in that they share the same secret key. Another approach is to use a public key system which can achieve the same result without actually having to share the same secret key. Again the P1 and P2 parameter bytes are used to reference the algorithm and secret data in the card.

Internal Authentication

This command completes the bilateral authentication in that the IFD checks the authentication of the card. In this case the random data is sent to the IC card by the IFD. The card then replies with the enciphered version of the random data. The IFD can check this cipher to prove the authenticity of the card.

Select File

The inter industry commands defined in the draft standard are all effectively operations upon a file. It is the purpose of this command to select the relevant file prior to the necessary operation. The file remains selected until another invocation of the select file command. The file may be referenced either as a path description (discussed previously) or as a file name. Within the command header the P1 and P2 parameter bytes are used to select which addressing option is being used. The data body of the command then carries the information necessary to select the required file.

Read Binary

The read binary command is used to read data directly from the selected EF file. The P1 and P2 parameter bytes are used to choose the offset from the start of the file for the first byte to be read. The Le byte in the data body of the command is used to define the number of bytes to be read. The main point to notice here relates to the data structure of a particular file. Quite clearly one cannot mix data stored in binary format with that recorded in a structured record format. A read binary command applied to a file stored in record structure would result in formatting information being mixed in with the data.

Read record

This command is used to read one or more records from an EF file. Normally the file would be selected with a select file command. However it is possible with this command to use a short EF identifier to select the particular file required. The P1 and P2 parameter bytes are used to establish the protocol of which record is accessed. It is also possible to read from a defined record until the end of the file. The Le byte in the data body of the command is used to define the total number of bytes to be read. This command of course should be rejected if the selected file is not stored in a record format.

Get Response

The T = 0 communication protocol has a number of limitations compared with the newer T = 1 protocol. For instance the T=0 protocol does not allow data to be sent in both directions as part of one command. The Get response function allows you to obtain response data generated as part of a command which also contains data as part of the command, whilst using the T=0 protocol. This command is initiated by the IFD. The Get response command belongs to the transmission oriented inter industry commands.

Envelope

This is the second command belonging to the transmission oriented inter industry commands. The envelope command may be used to overcome the lack of a chaining facility in the T=0 communication protocol. Accordingly it allows the IFD to assemble a command and data into a number of envelopes where the total data may exceed 255 bytes which is the normal limit of the T=0 communication protocol data transmission from the IFD to the ICC. Again this command is initialised by the IFD and is really only appropriate for the T=0 communication protocol. The concept of chaining using the T=1 communication protocol has been described previously.

Write Binary

The write binary is the complementary command to read binary. This command is used to write data into an EF file in an unstructured way (i.e not in a record format). The relevant file should previously have been selected by a select file command.

The actual physical writing of data to the memory of an ICC can be quite a complex operation. The process differs between EPROM and EEPROM memory. In this tutorial we have largely ignored the EPROM memory which requires the IFD to supply the memory programming voltage to the Vpp connector. This voltage varies (significantly) between the different chips which is why the necessary information must be contained within the answer to reset (ATR) interface bytes. The EEPROM devices generate the higher voltage required within the chip. It is also necessary for the correct timing sequence to be generated for the memory write operation. This operation typically takes 5mS. An erase operation also takes about 5mS. Some ICC devices have a page operation (typically 32 bytes) when the write and erase operation may be applied to a page at a time. Hence the writing of 32 bytes in this case will only take 5mS. Typically chips with EEPROM memory also allow an overwrite function. When the erase state of the memory is the `1' condition then this amounts to a logical `AND' operation. If the erase state is a `O' condition the overwrite operation amounts to a logical `OR' operation. Therefore a complete write operation may iinvolve two steps, an erase followed by an overwrite. All of these processes should be transparent to the application programmer.

Write Record

This is the complimentary function to `Read Record'. The command operates similarly to the read record, where the P1 and P2 parameter bytes are used to define the required record in the EF file. The command also allows the EF file to be identified by a short EF identifier which will override the currently selected file. The Le byte in the data body of the command is used to set the length of the data block. The command allows one of two write modes, append and update.

The Append operation adds a new record to the end of the file whilst the update is used to rewrite a defined record. This command should of course be rejected when the selected file is not structured as a file of records.

This completes our discussion of the Inter - Industry commands. As we have previously mentioned these commands are still subject to further debate in the ISO forum. Further more the commands are at some variance with those being standardised by ETSI. These problems are further compounded by the confusion over the role of security within the ISO standards and this will be our topic of conversation for next month.

Next month: Security and the Smart Card

David Everett

Smart Card Tutorial - Part 9

First Published in May 1993

Security and the Smart Card

It is really curious that we have managed to go so far in our tutorial without any serious consideration of security. It is an implicit assumption of most commentators that Smart Cards are secure devices and yet no real thought is applied to the basic architecture of the Smart Card. Security is an all pervasive attribute and must start at conception. We are in danger of getting it all wrong and one hopes that it is still not to late to unpick just a little of the emerging standards to put them on a sounder security basis.

The ubiquitous PC is a good starting point to look at security or rather the lack of it. When the PC's first started to emerge back in the 70s the requirement for security really was not there. The world was structured on the concept of central mainframe computers which in a way achieved better security through procedural operations than we see today where of course you have no control over the users who may well be operating in totally unattended environments. Back in those heady days the enthusiastic band of experimenters were more concerned with the benefits that might be obtained from using the little computer on the desk than the security requirements that are rather more obvious today.

Even the launch of the IBM PC in 1983 gave considerable emphasis to home use and the likely need to play games. Wordprocessing and spread sheets were still in their infancy (yes only 10 years ago).

The IBM PC and its disk operating system (DOS) is a good starting point for our discussion. This operating system was supplied by Microsoft and today in its newer variation is often referred to as MS-DOS. The architecture of the PC was based on a simple but none the less effective architecture as shown in fig. 1.

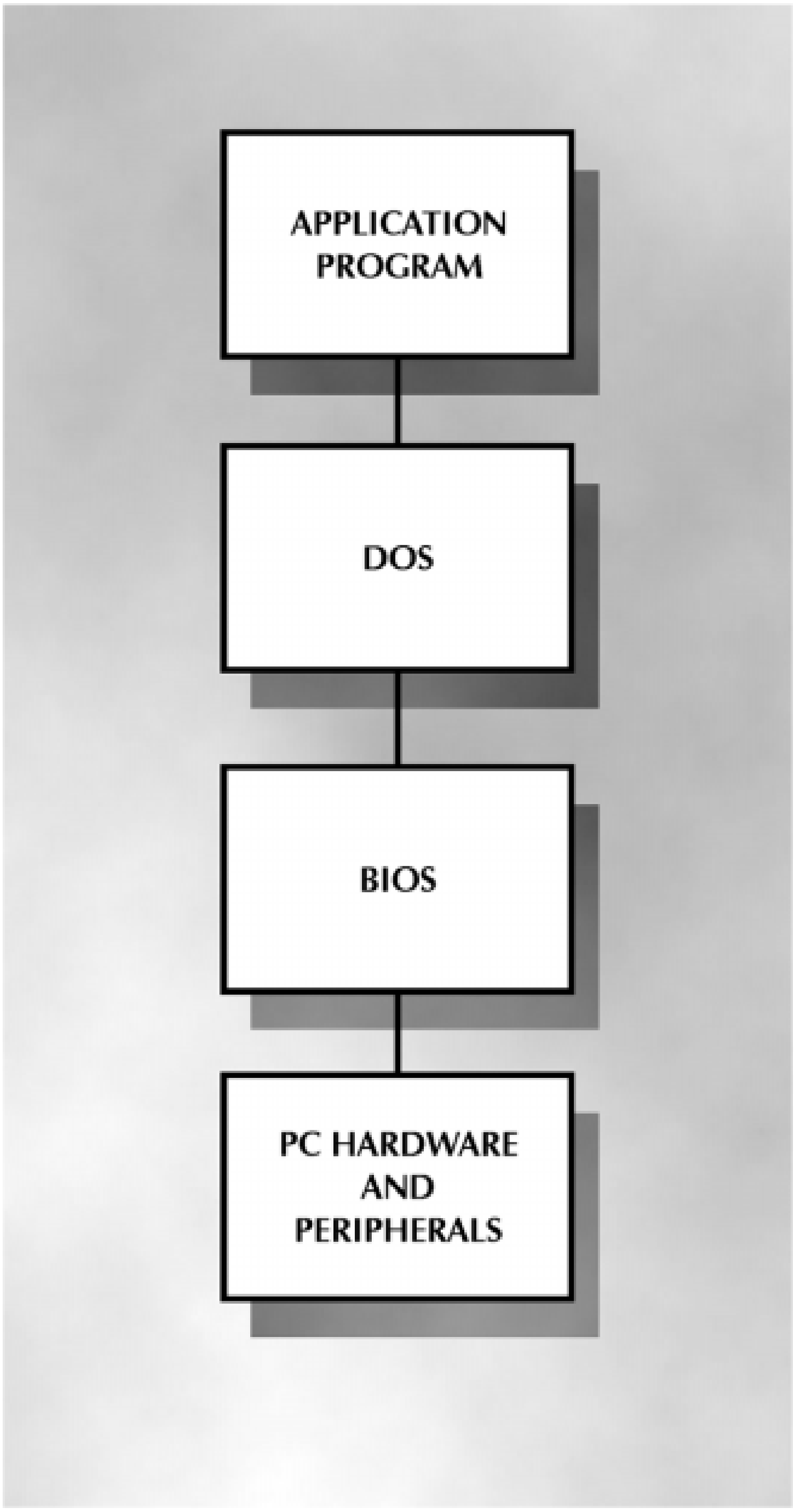


Figure 1: Basic PC Architecture

This is a three tiered software hierarchy that operates above the hardware. The BIOS (Basic Input/Output System) is the primary interface between the software and the hardware. This allows a hardware independent interface between the BIOS and DOS. The disk operating system offers the application a hardware independent interface. The separation between the DOS and BIOS can be continuously upgraded without changing the hardware or BIOS. Conversely the BIOS is fixed in a particular PC implementation and is normally provided as a ROM chip on the PC processors main motherboard.

In theory an application program should pass all its requests for peripheral services to the DOS program. This in terms translates the request into BIOS commands. The BIOS further translates these commands into direct interaction with the hardware devices.

In practice an application program once executing can totally ignore the DOS and BIOS and can interact with the hardware directly. This is often done to improve efficiency for applications such as communications where the application strives for maximum efficiency. The generality of the DOS and BIOS will not achieve the best performance. The particular point we need to notice at this stage is that the application is allowed direct access to the hardware including the disk drives which contain all the application data. Clearly at this stage any concept of security is a figment of the imagination. Neither the DOS or the BIOS include any security features.

Security requirements

Before we compare the architecture of a Smart Card with the PC let us stop and think about the security requirement that we might expect of such a device,

Access Control

Perhaps this is the starting point for any storage and processing system. The authorised user wishes to be assured that only he can read and modify his personal data. In other words he wants controls to prevent unauthorised access to his data (or programs). In normal security terms we would sub-divide this into two types of access,

- logical access control
- physical access control

Logical access control concerns such familiar principles as password checking or the more sophisticated cryptographic mechanisms for authentication. Physical access control relates to the difficulty of a perpetrator physically breaking into the store of data. For example connecting wires to the disk drive directly and bypassing the rest of the computer completely.

Authentication and tamper resistance

The principle security service we are concerned with at this stage are authentication and tamper resistance. Authentication may relate to source authentication (i.e confirmation of the identity of the source of communication) or peer entity authentication which assures one entity of the purported identity of the other correspondent. Tamper resistance is that fascinating feat of engineering that makes a device resistant to physical attack (proof against attack would be deemed an illusionary objective). As you might expect it is in this particularly difficult area that the IC chip offers significant benefits. It is probably true to say that it is also this objective that has been the most difficult to achieve adequately with conventional cryptographic equipment.

Data Integrity

For many applications and particulary in the financial world the preservation of data integrity is the principle security requirement. Here we are concerned with thwarting any event that results in the unauthorised tampering of the data. This includes not only modification of data but also addition or deletion of data.

Confidentiality

The need to preserve secrecy of data occurs in many applications. This relates both to the storage of data and the transmission of data. Where any cryptographic mechanisms are employed it is of course necessary to preserve the secret keys from compromise.

Non- Repudiation

Non - repudiation relates to that security service which ensures that a correspondent in some data interchange cannot subsequently deny his actions. Where trusted entities are communicating this facility is not required.

It is clear however that in most cases this is not the practical case and there is invariably a need to be able to resolve disputes between the various parties involved in an interchange.

Audit Services

In the ideal environment the application of security mechanisms should be transparent to the users. This results in some difficulty in being assured that security controls are operating correctly. It is not too different to that situation where you jump out of an aeroplane with a parachute. Its only then that you find out if it really works.

The correct design of audit controls and mechanisms can enable the system providers to be assured of the correct operation of the various security services.

Security mechanisms

There are a wide variety of security mechanisms available to the designer. The most important mechanisms are based on the use of cryptographic algorithms. By this means it is possible to create services for,

- Authentication
- Data Integrity
- Confidentiality
- Non - repudiation

In some cases a single mechanism can provide a number of security services. For instance a digital signature can offer data integrity with source authentication and non - repudiation.

The important point to note at this stage is that all these cryptographic controls involve key management. This requires the secure distribution of cryptographic keys into the various entities and the need for these entities to provide a tamper resistant environment.

Secure Smart Card Architecture

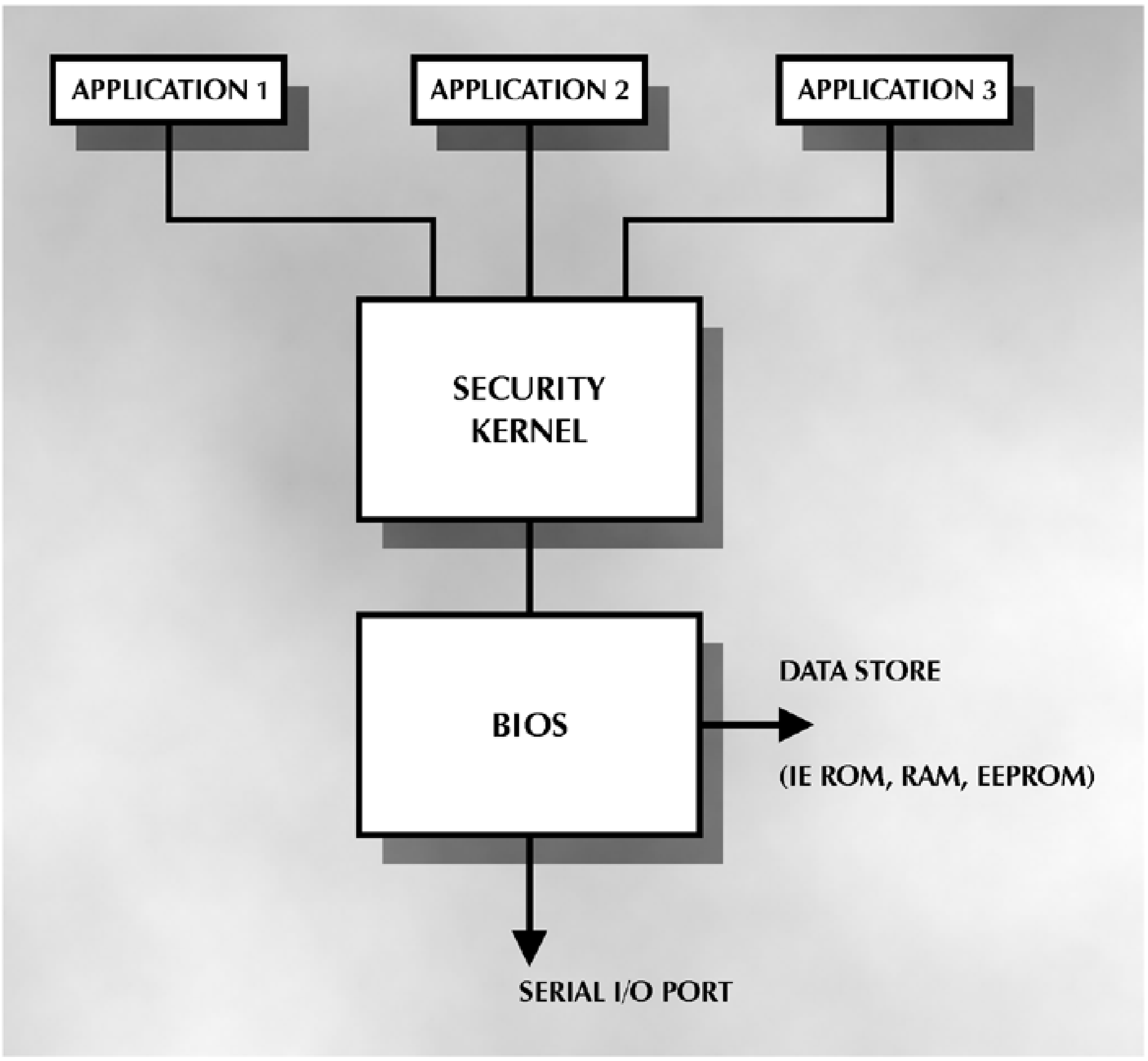


Fig. 2. A Secure Smart Card Architecture.

We can now examine the architecture of a Smart Card to see how we might create the necessary secure environment. In figure 2 we show a revised architecture to that for the PC.

The concept of the BIOS has been preserved to keep the interface between the software and the hardware. It is the introduction of the security kernel that is the basis of the real controls. There is really nothing new in these ideas for they have been implemented in large scale computers where it has been necessary to invoke the kind of security service we have discussed previously. The principle features of this kernel design are simplicity itself:

1) No application can take control of the processor in an unrestrained way

2) The mapping of data between the applications and the data store is enforced

3) Each application provides and operates its own security controls.

In terms of a hardware design for a Smart Card these principles can easily be achieved. Limiting the application can be through the software in the security kernel, for instance one could imagine an interpreter interface where all resource requests are checked against a rights access matrix. It is also possible to achieve control by special hardware in the microcontroller where an application can be constrained from accessing the EEPROM directly. These tools can ensure that an application can only access the data to which it is authorised and in the prescribed way.

The other important feature is to preserve security segregation between the various applications. This means that each application should implement its own controls for authentication, data integrity, access control etc. It may be that common mechanisms are provided in the kernel but the management of the keys should be part of the application control.

Next month. The design of a secure Smart Card will be developed further.

David Everett

Smart Card Tutorial - part 10

First Published in June 1993

Security from the Bottom End.

Previously we took a look at the security of the Smart Card from a top down point of view. In other words we looked at the principles that we were trying to achieve without delving into the practicalities. This month we are going to start at the other end, looking at some of the practicalities to see what can be achieved. This bottom up approach should allow us to meet somewhere in the middle. This is a compromise between what is required and what can be achieved.

In order to consider security further we need to recap on the basic components of the chip in the Smart Card. This architecture is shown in fig. 1. The processor has four peripherals:

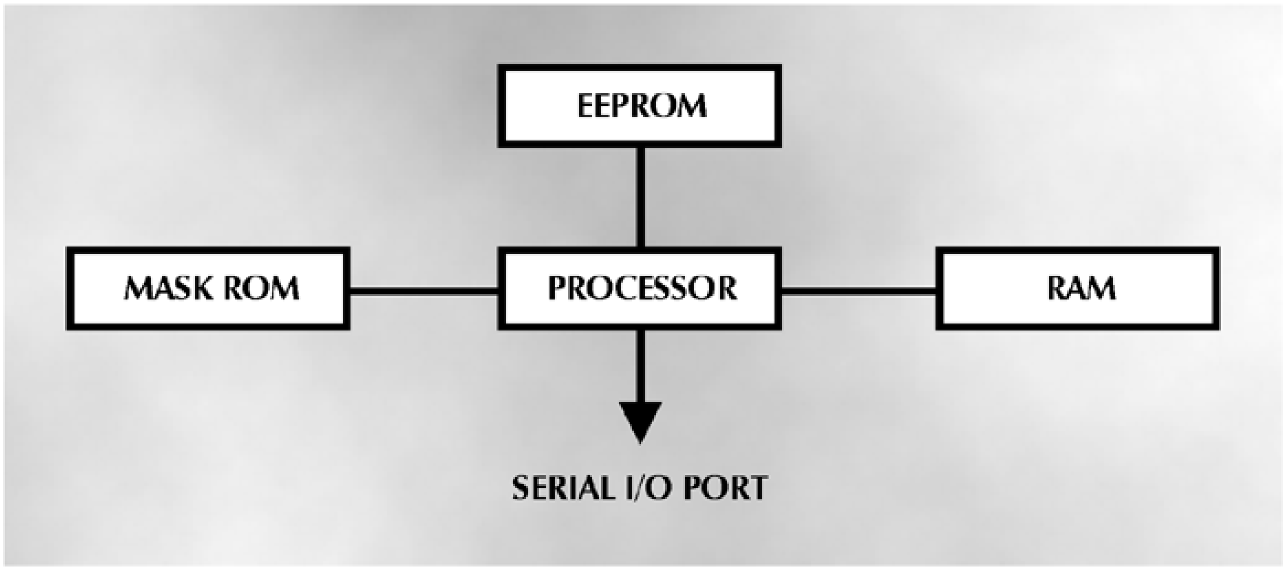


Figure 1: Smart Card Chip Architecture

- MASK ROM
- EEPROM
- RAM
- SERIAL I/O PORT

The mask ROM contains the operating system of the chip and is made as part of the chip fabrication process. This memory is read only and cannot be changed once the chip is made. The ROM may contain programs and data but in both cases the code and data are constant for all time. By the very process that the chips are made it is not practical to have any form of unique code or data in ROM. Thus the ROM memory is constant for a batch of chips (thousands). Each wafer at the end of the manufacturing process results in the die (apart from fabrication failures) looking identical.

The EEPROM memory is the non-volatile storage area of the chip that allows data to be written and read under program control. This data is preserved even after the power to the chip is switched off. By writing data into the EEPROM we can give each chip a unique identity. The Smart Card chips from most semiconductor manufacturers have the facility to make parts of the EEPROM memory `write once only'. This is sometimes called OTP (One Time Programmable) or occasionally as EPROM memory in the sense that it cannot be overwritten. The latter term is ambiguous in that although EPROM memory requires ultra violet light for erasure, in the general sense the memory cells are always capable of being set to the final state. Thus if the initial state is all `ones' then any bit can be overwritten to `zero'. If this situation is allowed to arise then in some circumstances you may be subject to a security violation. Under these conditions going from a `1' to a `0' must increase the security for every bit used. A reverse situation may allow an attacker to decrease the security by over writing a `1' to a `0' which is an inherently possible process.

The random access memory (RAM) forms the memory working space to be used by the processor whilst executing programs either in ROM or EEPROM. This memory is volatile and all data will be lost (there are some security subtleties here that we will return to in a subsequent part) when the power to the chip is removed.

This RAM is no different in concept to that contained in our PC. However there is some difference in the amount of memory available. The modern PC usually starts at 1 million bytes and commonly has 4MB or more. The lowly Smart Card chip rarely exceeds 256 bytes. We mentioned previously that this is due to the square area of silicon taken by the RAM cells and the need to limit the size of the die for both cost and reliability considerations. Clearly the processor has total read/write control of the RAM. It is also important to note that the total RAM space is unlikely to be available to the application. At the very best it is necessary to invoke a stack memory area for the processor to transfer control between the various software modules and to handle the interrupt structure of the processor.

The serial I/O port should be considered as just another peripheral to the processor which may be read and written under software control. The most important point to notice here is that the hardware sophistication often found on general purpose microprocessors has been removed to optimize the space available on the silicon. Thus the ubiquitous UART (Universal Asynchronous Receiver Transmission) which buffers bytes of data to and from the serial port is replaced by a single register that the programmer must manage on a bit by bit basis. Further more the timing of data transmission which is handled by the UART must now be managed by the program in the Smart Card.

For the purpose of our security analysis we will now consider two application scenarios. In the first case we will look at the Smart Card as a file management system as considered under ISO 7816-4. Then we will develop the situation further and look at the problems of managing two application programs in the IC.

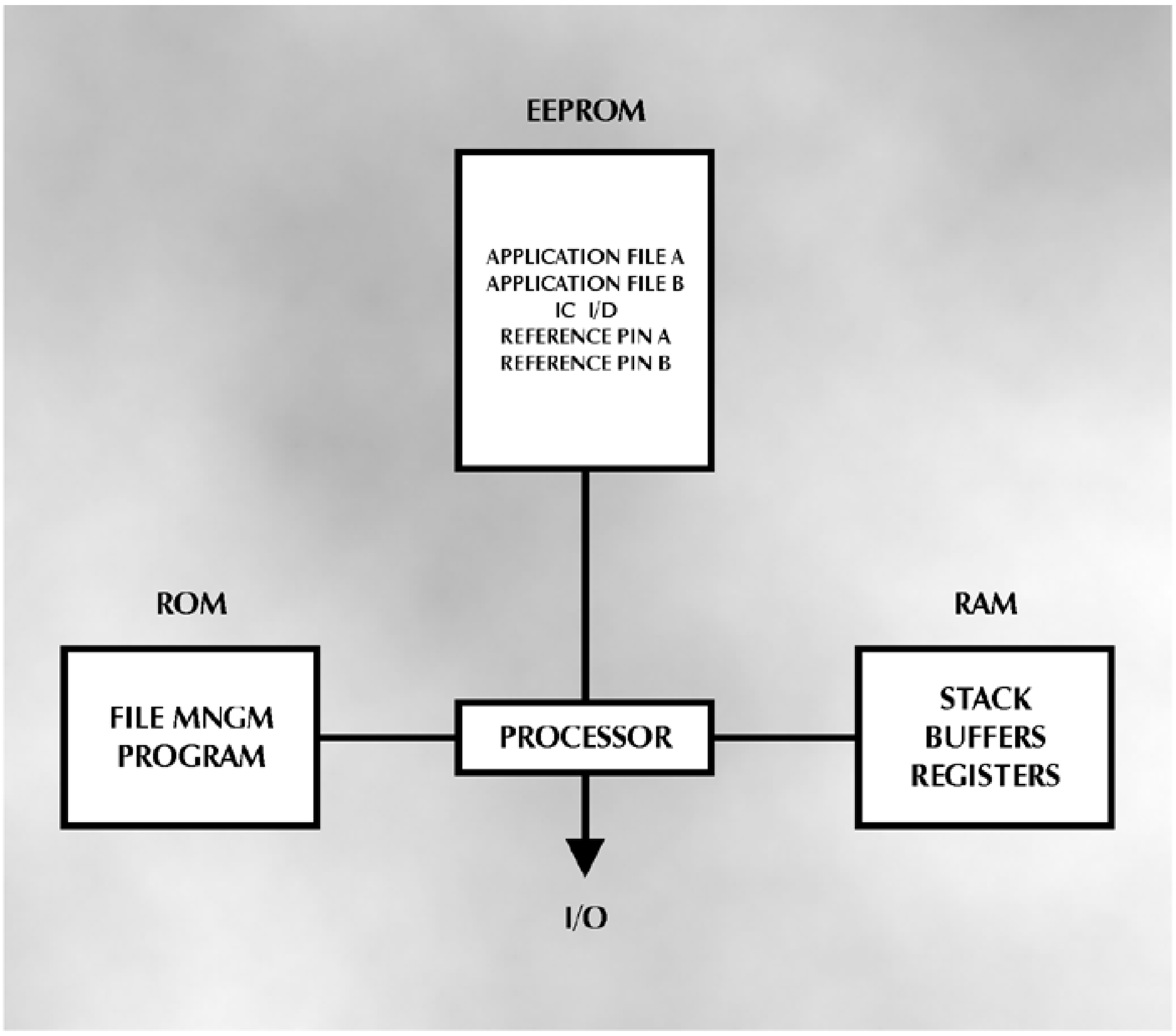


Figure 2: The Smart Card as a File Management System

In fig.2 we show the arrangement of programs and data for the Smart Card used as a file management system. We will simplistically consider two applications each with a file of data. We will also assume that these applications wish to control access to the data for authorised users only. It is important to note that the terminal acts as the application driver and completes the security link. Let us now consider that the Smart Card is brought into contact with a terminal containing the application as shown in fig. 3. In this discussion we will ignore the electrical and communication protocol handling and will assume it meets the ISO standard.

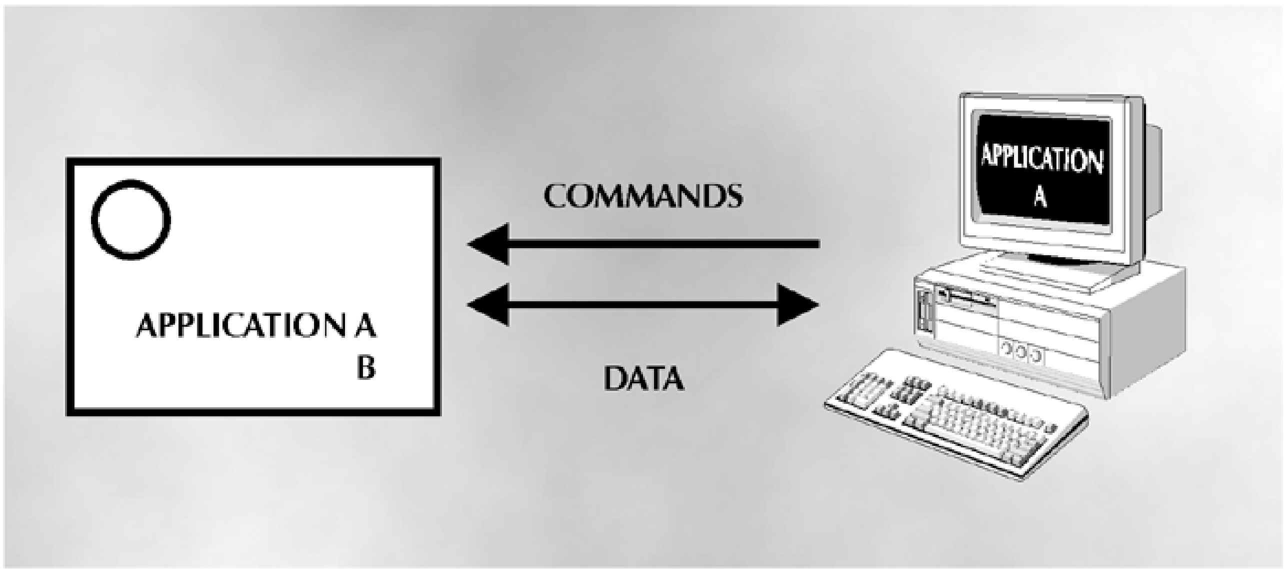


Figure 3: A Simple File Management Application

From the terminal's point of view there are four primary steps in the process of executing the application,

- Select the application in the card
- Prove the authorisation of the terminal user
- Read/write the application data
- De select the application (e.g power off)

In this very simple example we are only considering PINs as our security tool and the authorisation is therefore that of the terminal user (which may be delegated to the terminal by the application provider)

The application in the terminal thus proceeds to select the application using the commands of ISO 7816-4 as discussed previously (select file; verify; read/write).

Even in this simple example we run into problems straight away. Does each application have a separate PIN? From a security point of view it is clear that this must be the case and yet this contradicts the often held approach (with its obvious practicality) that this should be a single PIN for the card. There is a second problem even more fundamental than the first. How does the terminal know that the card is genuine? Giving a yes/no to the verify command is totally inadequate and hence the need for the authentication command. This allows the terminal to check the authenticity of the card but requires both the terminal and card to share the appropriate cryptographic mechanisms.

However it is clear that sufficient functionality exists to control these applications separately. Here the operating system is in control and can easily restrict access to the application data to authorised users in the sense that the correct PIN is provided). The application program in the terminal has no access to the data in the EEPROM directly and must invoke the commands available in the MASK ROM.

Let us now consider the more interesting case where there are two application programs in the EEPROM as shown in fig. 4. Now the security game changes because the processor effectively transfers control to a program running in EEPROM. In the general case (some IC chips can constrain the memory partitioning; see vol. 1, No 1) the processor can read and write any data in the EEPROM whether it belongs to its own application or another. What this means is that a particular application must be restricted from reading and writing data in the EEPROM. All data accesses must be referred to a program that executes from the operating system in the MASK ROM. By this means the operating system can assure the correct partitioning of the data to its own application. Whether this is achieved by software (i.e an interpreter type of approach) or hardware control of the memory accesses results in a more sophisticated view of the architecture of the ICC.

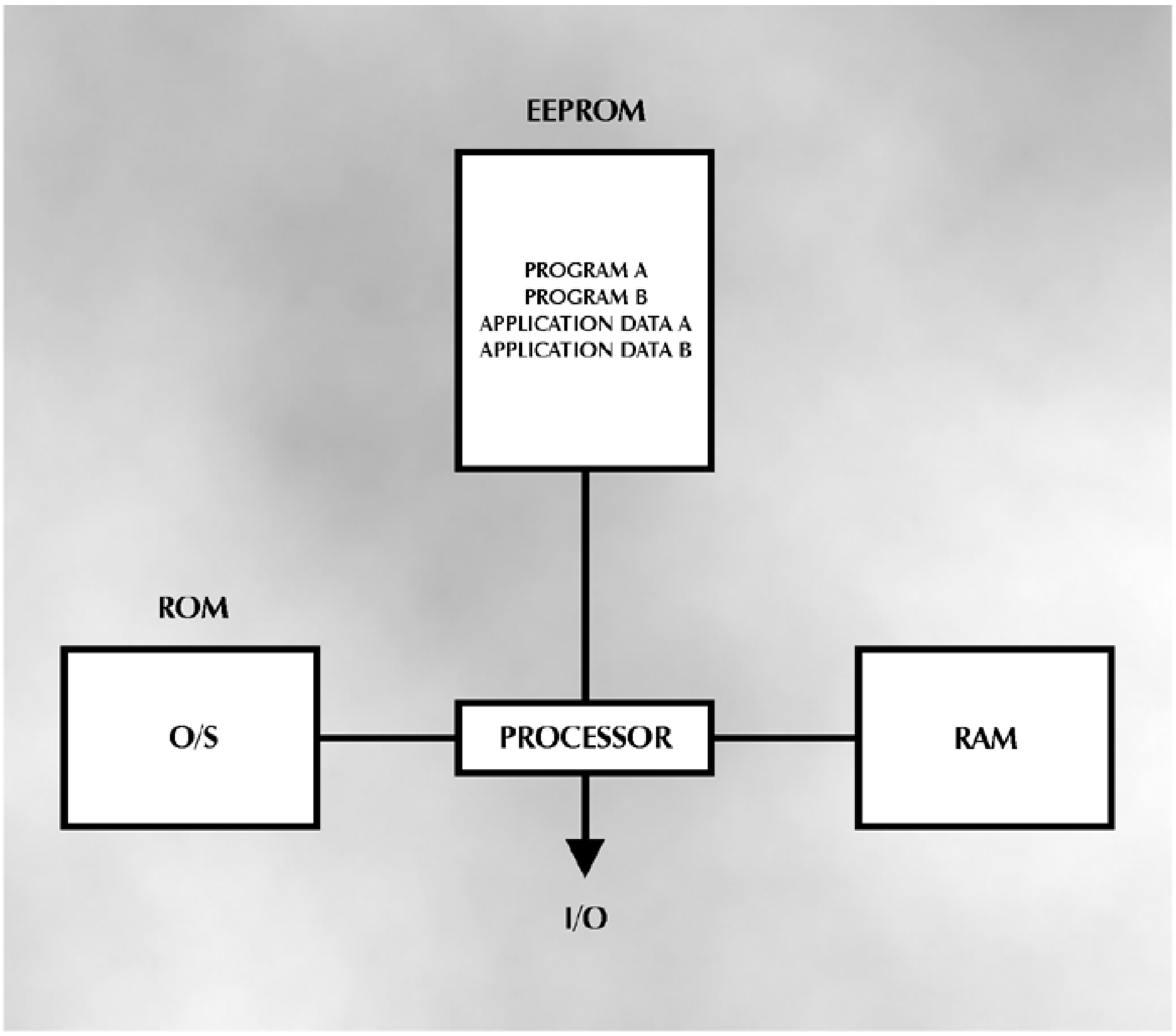


Figure 4: A Muilti Application Environment

We have shown in this part of the tutorial that there is a fundamental security difference between a file management structure (as envisaged in ISO 7816-4) and the more general case of a multi application environment. We have also made the point that a PIN check (supplied by the terminal) by an IC card is a one way process which does not take account of the authentication of the card itself. This is clearly not acceptable in the majority of applications and requires therefore the additional process of the terminal authenticating the card. This requires an additional overhead of cryptographic mechanisms and the appropriate key management hierarchy.

In a subsequent part we will explore the life cycle of an IC card from a security point of view including the implications of cryptographic key management. We will also attempt to answer that difficult question `Is a Smart Card secure?'

Next month. The Smart Card development environment.

David Everett

Smart Card Tutorial - Part 11

First Published in July 1993

The Development Environment

I can not help thinking that the semiconductor manufacturers have missed a trick when it comes to the ICC development environment. It is not easy for application developers to independently create their own schemes due to the somewhat closed shop approach of the majority of semiconductor manufacturers and card fabricators.

In the mid 70's the microprocessor was almost unheard of, then suddenly over just a few years the world was inundated with microprocessors for everything. A position that today we all take for granted. The course of events at the time seemed so obvious with the aggressive marketing of specifications, application notes and simple evaluation or development kits by the major semiconductor houses. At the time I remember the superb development packs produced by Motorola for their 6800 microprocessor. But today we seem to have the reverse, getting information is worse than getting blood out of a stone. With one particular semiconductor manufacturer (not Motorola) I was personally bounced from office to office for a week just to find out the die size of a chip in current production. After numerous faxes and telephone calls the question was never answered. Do they really expect me to have to get out a ruler. (Note: any school laboratory is perfectly capable of removing the resins covering the die assembled as part of the chip micromodule).

If I assume that the reaction to my approach to the suppliers is typical then new potential users must be finding it difficult just to obtain the basic chip specifications. Smart Card News decided to produce an article comparing the technical specifications of the major semiconductor suppliers offerings. After one month and numerous reminders some companies have still failed to supply even the basic information.

In terms of development hardware and software whilst most chip manufacturers have some form of offering they are often hidden away to deter all but the toughest enthusiasts. The problem seems to be the distribution route for Smart Card chips. Those organisations undertaking a card fabrication role are the intermediary between the user (here the application provider) and the chip manufacturer. One of the major roles of the fabricator is to develop and supply complete application systems and not surprisingly they are often less interested in the do it yourself brigade. However there is a need for both and it seems clear to me that the first chip manufacturer to get his act together will probably lead the field in what is becoming a rapidly emerging market with enormous potential.

In this part of the tutorial we will examine the various approaches to application development based on tools that are available in the marketplace. In separate articles we will review some of the standard offerings in more detail.

We have often discussed the basic components of the IC chip (reproduced in fig. 1). For now we need to remember that the core operating system resides in the ROM memory and this will be executed on reset. The EEPROM memory will hold the application data and optionally additional application programs. It should be noted here that we have tended to ignore the use of non volatile EPROM memory. This memory is limited to write once and is therefor not as flexible as EEPROM memory. However it can be used to store application programs and to store data that is not required to change and for which there is sufficient capacity to meet the application requirement. In our discussions readers may consider the use of EPROM memory as appropriate in some application scenarios. The advantage of EPROM memory is that it occupies a smaller area than the equivalent EEPROM capacity. Accordingly this results in a lower cost device.

The RAM memory is the working space used by the application whether executing in ROM or EEPROM. From a developers point of view it is important to realize that the ROM memory is fabricated as part of the chip manufacturing process. This requires a more extensive development and results in a typical turnaround of about 3 or 4 months from the semiconductor house for the receipt of working samples. One manufacturer (Atmel) has recently produced a new chip that has 16K bytes of EEPROM where 8K bytes are used for the operating system instead of the ROM memory. This memory can be programmed as the last step in the chip manufacturing process and therefore enables a more rapid turnaround time.

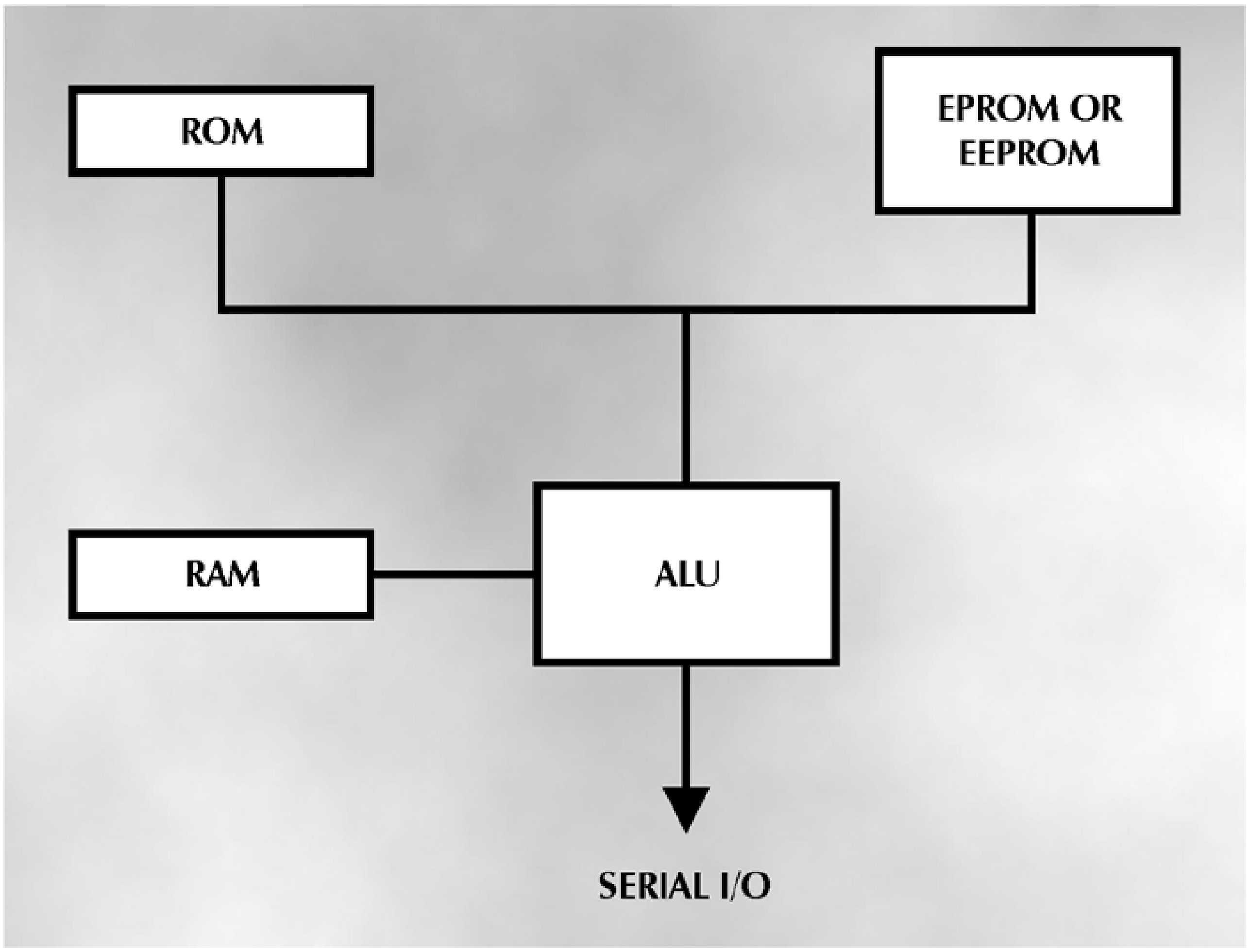


Figure 1: The IC Basic Architecture

There are broadly speaking three development options that may be pursued, shown here in increasing order of complexity,

- Use an existing application in the ROM, where the EEPROM is used for the management of application data.
- Add an application program to the EEPROM to work in conjunction with the ROM program.
- Develop an application for the ROM.

In all cases there is a need to develop a matching application in the terminal to which the IC card will be connected. Each of these development options will be considered in turn .

a) Suitable application already provided in chip ROM

This is the easiest entry point for an application developer assuming that an IC card can be obtained for which an appropriate program already exists in the ROM. A number of suppliers provide IC cards in this format including,

- GEMFILE (from GEMPLUS)

- OSCAR (from GIS)

in both cases sited here the chip is already programmed with a file management application along the lines of the emerging ISO 7816 - 4 proposed standard. Conformance to the proposed standard is not a matter that we need to discuss here. These IC cards are designed for general purpose evaluation and development. Here the task of the developer is to produce the application in the terminal device that will be used in conjunction with the card. It is clear of course that the application may involve more than just the terminal and will often include other components such as a host computer system. When we refer to the terminal application development it is intended to include the application system into which the IC card operates.

We have assumed that the development process follows conventional design principles with the definition and production of the necessary specifications along the following lines,

- Technical requirements specification
- Functional specification
- Architecture specification
- Component specification
- Test specification

In this part of the tutorial we are concerned far more with the tools available for managing development. We will leave the rigors of software specification, design and development for another occasion. Here in particular we would like to evaluate an existing component to assess its viability for incorporation into the business application. In fig. 2 we show a typical set up for evaluating the IC card. In this tutorial we have always used the ubiquitous PC as our core processing engine. This should not be taken to mean that other processing systems are excluded but only that we are reflecting the simplest and most readily available development tools.

Now the basic standards start to become important. We would expect the IC card to conform to ISO 7816 - 1/2 in terms of physical size and location of the contacts. Also we would hope that the card would conform to ISO 7816 - 3 in terms of its electrical and signal characteristics. The main point to watch here is the communications protocol for the IC card. The T=0 protocol is well tried and tested whilst the T=1 protocol is somewhat newer and subject to options. Perhaps the least explored area is protocol type selection where the IC card is capable of operating with more than one communication protocol. I am not aware of any commercial product that is capable of handling this complexity. In fact it is far more appropriate for the terminal to handle different protocols say both T = 0 and T = 1 whilst the card may handle only one of these protocols.

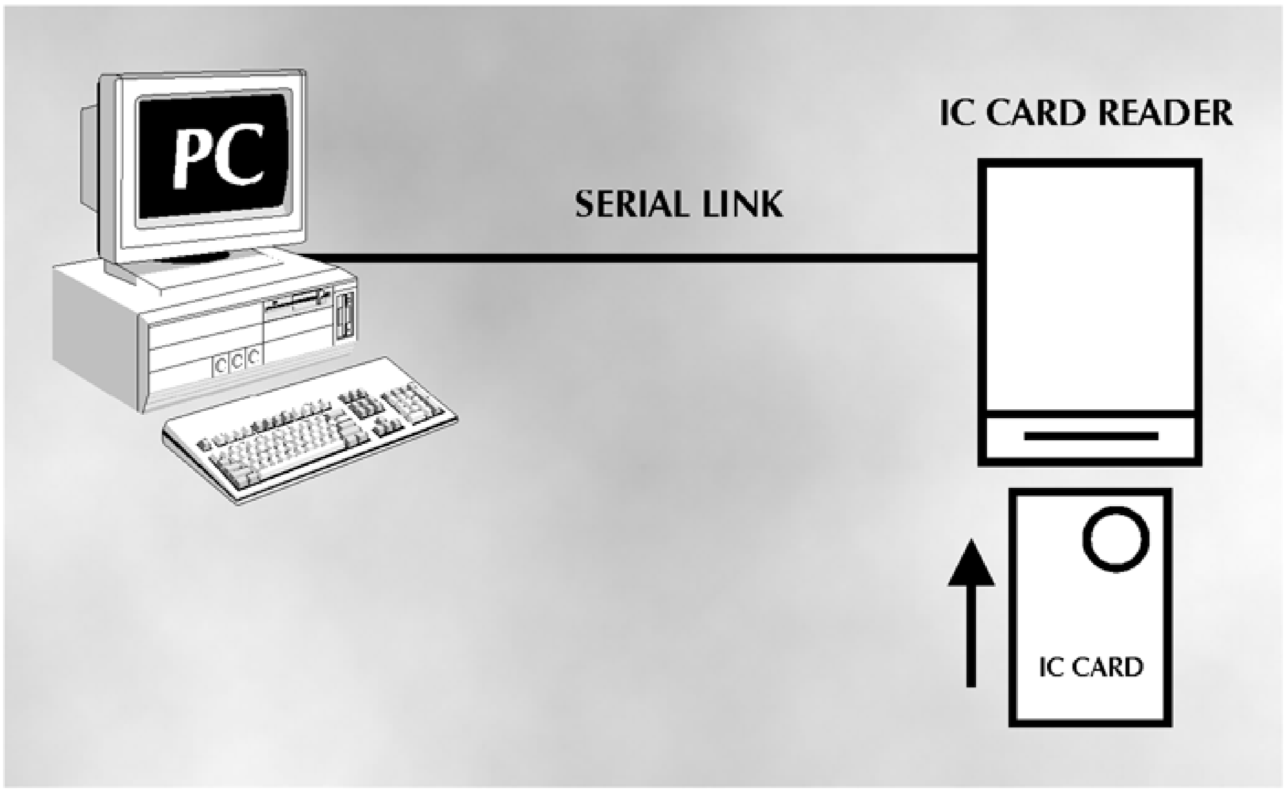


Figure 2: Basic Card Evaluation Configuration

Accordingly the basic set up for evaluating the IC card product is a PC on which an evaluation software package has been installed and an IC card reader (conforming to ISO 7816 - 1/2/3) which is usually connected to one of the serial ports on the PC.

The evaluation package would normally contain a menu that allows the user to issue the allowable command range and to show the IC card response. Thus in general facilities are provided that allow the user to read and write data into the EEPROM memory. The full command range will be defined in the documentation that comes with the IC card specification. These commands should operate as per the ISO 7816 - 3 standard and are usually similar to the set of commands currently under discussion for the proposed ISO 7816 - 4 standard. We described these commands in an earlier part of the tutorial (part 8).

Now some applications may only need this basic file management system with its somewhat limited security capability. In this situation the task of the developer is to build the terminal application which can be developed using the same hardware configuration as shown in fig.2.

b) Develop an additional program to the ROM operating system

The next level of sophistication in terms of a development strategy would be to add an additional application program which can be executed from the EEPROM memory. Clearly the ROM operating system must be designed from the start to allow this form of development. The COS (Card Operating System) supplied by Gemplus as a standard product allow for this enhanced application.

The COS operating system includes the concept of filters. This is a means by which the operating system can transfer control to an application program that has been loaded into the EEPROM memory. The developer will need to design his application using the appropriate software tools (an assembler is a minimum requirement) for the particular chip that is used by the IC card. Having developed this application then the machine code may be written into the EEPROM memory using the standard write memory command which is provided by the ROM operating system.

The trick provided by the COS operating system is the ability to set a vectored address in a defined memory location. When the operating system is executing it will examine this address at the appropriate moment (e.g on receipt of a command from the serial port) and if the address has been set then control will be transferred to the new application program residing in the EEPROM memory. This allows the additional application to manage the commands received over the serial interface. The COS operating system allows some flexibility in the way these filters are managed. From a developers point of view this approach is attractive because it allows a very fast development path. Security may be enhanced but without having to wait for the delay in reprogramming (and developing) the mask ROM operating system. The only draw back to this style of development relates to the use of valuable EEPROM space for storing the application program. The adequacy of the security features is a function of the operating system. There is no inherent reason as to why this should not result in a secure development path. It must of course be accepted that the developer of the operating system will be looking for a return on his investment but for many situations this must still lead to a cost effective development path.

c) Develop a new ROM operating system

This is of course the most involved of all the development paths. It is readily apparent that this can be a non trivial task and for an operating system with the complexity of the COS operating system referred to earlier almost certainly accounts for many man years of development. However for those situations where card cost is an overriding factor then the development of a ROM operating system allows the use of the EEPROM to be minimal or perhaps avoided all together.

The development environment in this situation is considerably more complex and results in the need for more components as shown in fig. 3. The centre of the development kit is the chip emulation system. This is manufactured to contain the components of the chip in an accessible form. Thus returning to fig. 1 the developer, by means of the development workstation is able to access the memory components separately in a form of test mode. Thus the workstation can load an operating system into the ROM memory area (usually implemented using RAM) and instruct the chip to execute the program. Like any normal ICE (In Circuit Emulation system) the developer can set break point and debug his program. The chip emulation system and in particular the probe is manufactured to behave exactly like the single chip equivalent. Some times this is achieved by using a bond out version of the IC card chip where all the address and data busses are made available to the emulation system.

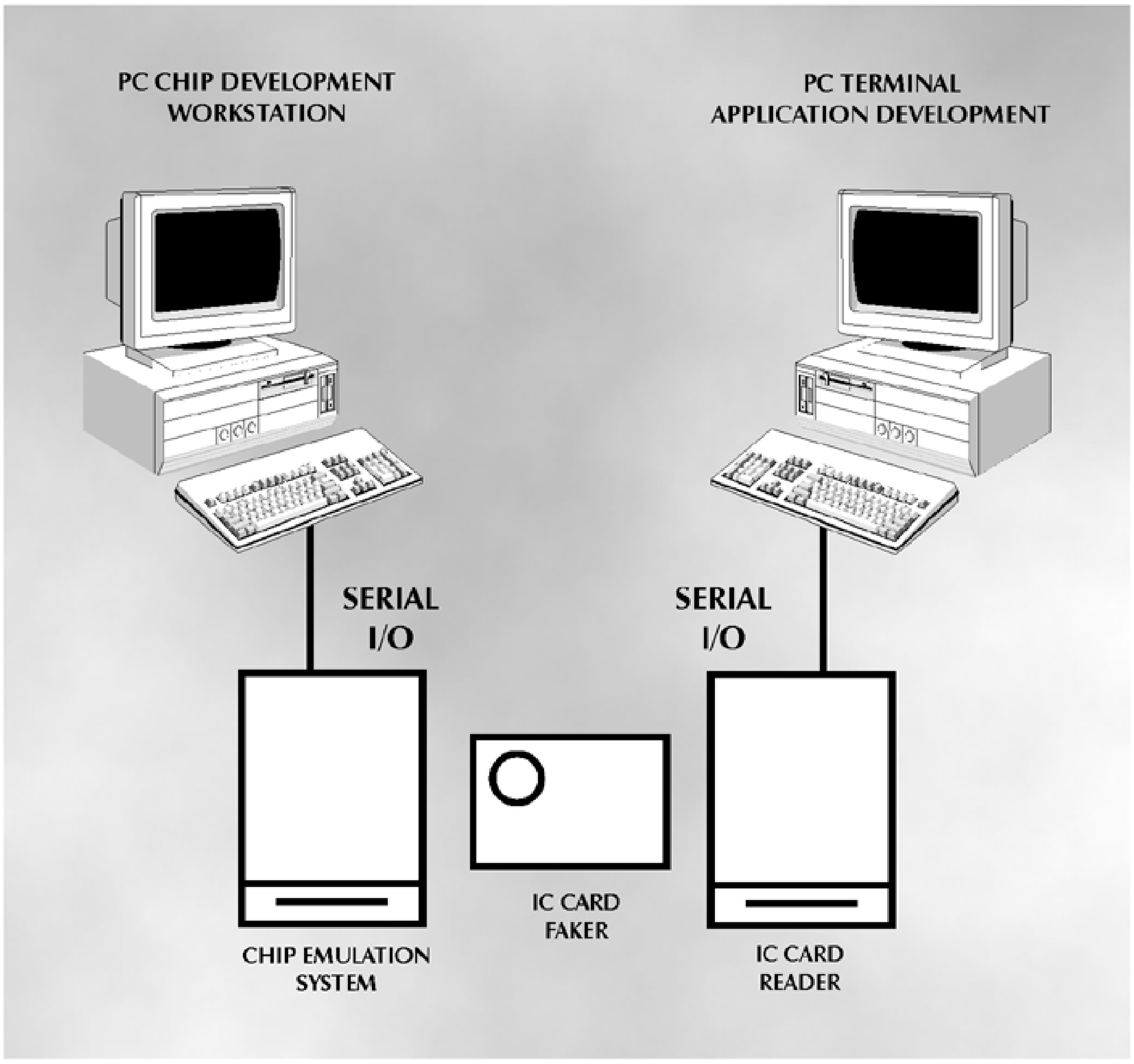


Figure 3: ICC Application Development Configuration

Having developed the new operating system in the emulator the total application can be tested by connecting a PC which contains the terminal end of the application. These two units are connected together by using what is sometimes called a faker card. That is a printed circuit board which has the same dimensions as an ISO 7816 -1/2 IC card at the connector end but which has a longer tail with a cable for connection to the emulator. The business end is plugged into a standard card reader (which can normally cope with the elongated tail) which is connected normally by a serial link to the PC terminal application.

By this means the developer can test the total application before having to manufacture the chip with the new operating system program. When the code has been tested as satisfactory in the emulation system then you can be pretty sure that when the chip is fabricated with the same code in the mask ROM that it will behave in the same way. This is not to suggest that the testing of the working sample produced by the semiconductor house can be avoided but only that one may reasonably expect success the first time round in the majority of cases where the initial emulation was fully evaluated.

David Everett

Next month: IC card security life cycle

Smart Card Tutorial - Part 11

First Published in July 1993

The Development Environment

I can not help thinking that the semiconductor manufacturers have missed a trick when it comes to the ICC development environment. It is not easy for application developers to independently create their own schemes due to the somewhat closed shop approach of the majority of semiconductor manufacturers and card fabricators.

In the mid 70's the microprocessor was almost unheard of, then suddenly over just a few years the world was inundated with microprocessors for everything. A position that today we all take for granted. The course of events at the time seemed so obvious with the aggressive marketing of specifications, application notes and simple evaluation or development kits by the major semiconductor houses. At the time I remember the superb development packs produced by Motorola for their 6800 microprocessor. But today we seem to have the reverse, getting information is worse than getting blood out of a stone. With one particular semiconductor manufacturer (not Motorola) I was personally bounced from office to office for a week just to find out the die size of a chip in current production. After numerous faxes and telephone calls the question was never answered. Do they really expect me to have to get out a ruler. (Note: any school laboratory is perfectly capable of removing the resins covering the die assembled as part of the chip micromodule).

If I assume that the reaction to my approach to the suppliers is typical then new potential users must be finding it difficult just to obtain the basic chip specifications. Smart Card News decided to produce an article comparing the technical specifications of the major semiconductor suppliers offerings. After one month and numerous reminders some companies have still failed to supply even the basic information.

In terms of development hardware and software whilst most chip manufacturers have some form of offering they are often hidden away to deter all but the toughest enthusiasts. The problem seems to be the distribution route for Smart Card chips. Those organisations undertaking a card fabrication role are the intermediary between the user (here the application provider) and the chip manufacturer. One of the major roles of the fabricator is to develop and supply complete application systems and not surprisingly they are often less interested in the do it yourself brigade. However there is a need for both and it seems clear to me that the first chip manufacturer to get his act together will probably lead the field in what is becoming a rapidly emerging market with enormous potential.

In this part of the tutorial we will examine the various approaches to application development based on tools that are available in the marketplace. In separate articles we will review some of the standard offerings in more detail.

We have often discussed the basic components of the IC chip (reproduced in fig. 1). For now we need to remember that the core operating system resides in the ROM memory and this will be executed on reset. The EEPROM memory will hold the application data and optionally additional application programs. It should be noted here that we have tended to ignore the use of non volatile EPROM memory. This memory is limited to write once and is therefor not as flexible as EEPROM memory. However it can be used to store application programs and to store data that is not required to change and for which there is sufficient capacity to meet the application requirement. In our discussions readers may consider the use of EPROM memory as appropriate in some application scenarios. The advantage of EPROM memory is that it occupies a smaller area than the equivalent EEPROM capacity. Accordingly this results in a lower cost device.

The RAM memory is the working space used by the application whether executing in ROM or EEPROM. From a developers point of view it is important to realize that the ROM memory is fabricated as part of the chip manufacturing process. This requires a more extensive development and results in a typical turnaround of about 3 or 4 months from the semiconductor house for the receipt of working samples. One manufacturer (Atmel) has recently produced a new chip that has 16K bytes of EEPROM where 8K bytes are used for the operating system instead of the ROM memory. This memory can be programmed as the last step in the chip manufacturing process and therefore enables a more rapid turnaround time.

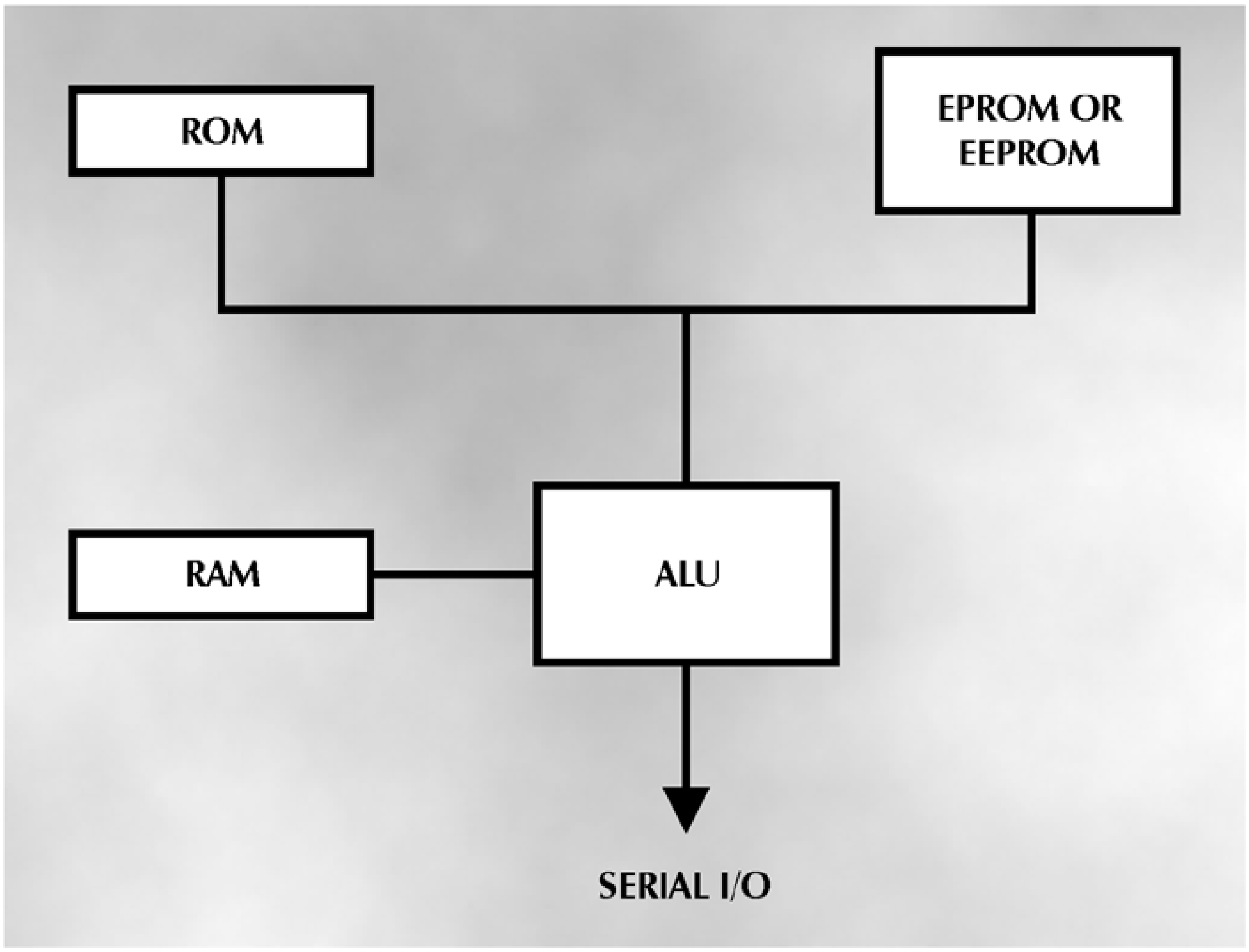


Figure 1: The IC Basic Architecture

There are broadly speaking three development options that may be pursued, shown here in increasing order of complexity,

- Use an existing application in the ROM, where the EEPROM is used for the management of application data.
- Add an application program to the EEPROM to work in conjunction with the ROM program.
- Develop an application for the ROM.

In all cases there is a need to develop a matching application in the terminal to which the IC card will be connected. Each of these development options will be considered in turn .

a) Suitable application already provided in chip ROM

This is the easiest entry point for an application developer assuming that an IC card can be obtained for which an appropriate program already exists in the ROM. A number of suppliers provide IC cards in this format including,

- GEMFILE (from GEMPLUS)

- OSCAR (from GIS)

in both cases sited here the chip is already programmed with a file management application along the lines of the emerging ISO 7816 - 4 proposed standard. Conformance to the proposed standard is not a matter that we need to discuss here. These IC cards are designed for general purpose evaluation and development. Here the task of the developer is to produce the application in the terminal device that will be used in conjunction with the card. It is clear of course that the application may involve more than just the terminal and will often include other components such as a host computer system. When we refer to the terminal application development it is intended to include the application system into which the IC card operates.

We have assumed that the development process follows conventional design principles with the definition and production of the necessary specifications along the following lines,

- Technical requirements specification
- Functional specification
- Architecture specification
- Component specification
- Test specification

In this part of the tutorial we are concerned far more with the tools available for managing development. We will leave the rigors of software specification, design and development for another occasion. Here in particular we would like to evaluate an existing component to assess its viability for incorporation into the business application. In fig. 2 we show a typical set up for evaluating the IC card. In this tutorial we have always used the ubiquitous PC as our core processing engine. This should not be taken to mean that other processing systems are excluded but only that we are reflecting the simplest and most readily available development tools.

Now the basic standards start to become important. We would expect the IC card to conform to ISO 7816 - 1/2 in terms of physical size and location of the contacts. Also we would hope that the card would conform to ISO 7816 - 3 in terms of its electrical and signal characteristics. The main point to watch here is the communications protocol for the IC card. The T=0 protocol is well tried and tested whilst the T=1 protocol is somewhat newer and subject to options. Perhaps the least explored area is protocol type selection where the IC card is capable of operating with more than one communication protocol. I am not aware of any commercial product that is capable of handling this complexity. In fact it is far more appropriate for the terminal to handle different protocols say both T = 0 and T = 1 whilst the card may handle only one of these protocols.

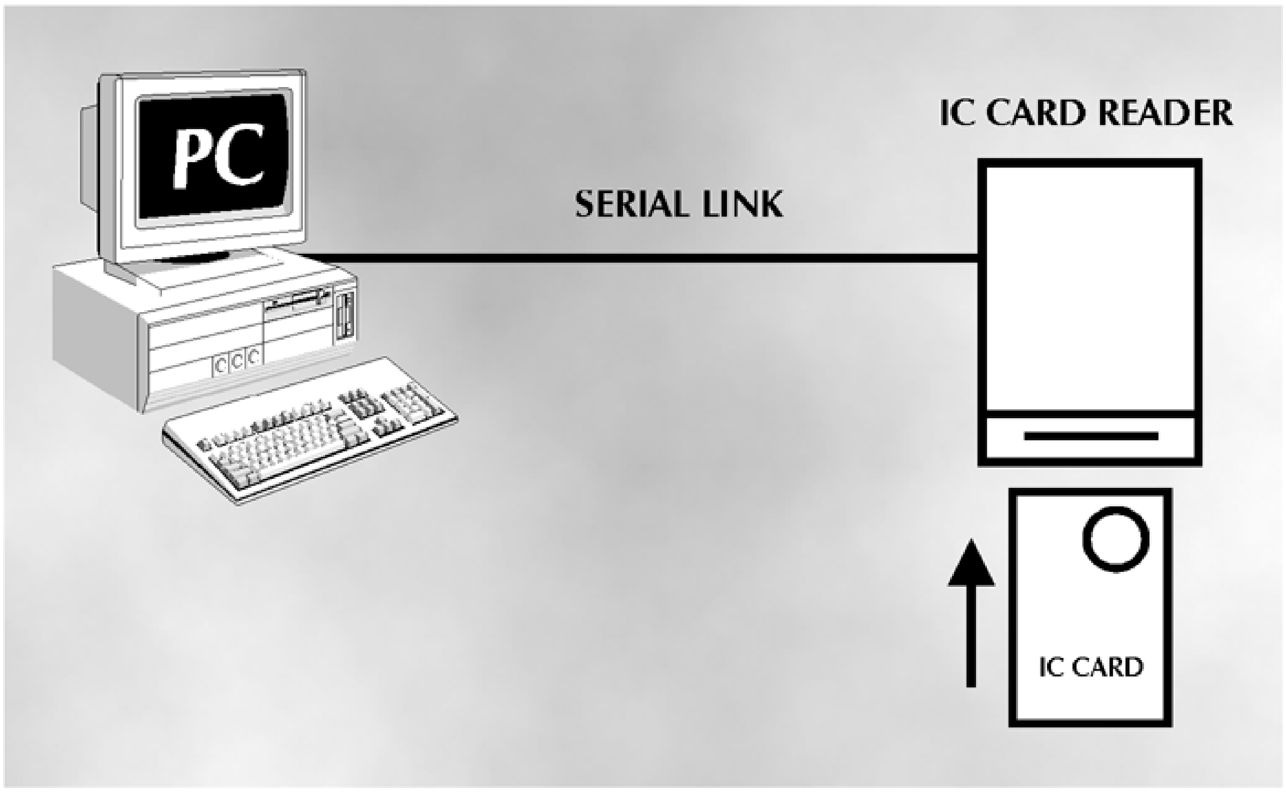


Figure 2: Basic Card Evaluation Configuration

Accordingly the basic set up for evaluating the IC card product is a PC on which an evaluation software package has been installed and an IC card reader (conforming to ISO 7816 - 1/2/3) which is usually connected to one of the serial ports on the PC.

The evaluation package would normally contain a menu that allows the user to issue the allowable command range and to show the IC card response. Thus in general facilities are provided that allow the user to read and write data into the EEPROM memory. The full command range will be defined in the documentation that comes with the IC card specification. These commands should operate as per the ISO 7816 - 3 standard and are usually similar to the set of commands currently under discussion for the proposed ISO 7816 - 4 standard. We described these commands in an earlier part of the tutorial (part 8).

Now some applications may only need this basic file management system with its somewhat limited security capability. In this situation the task of the developer is to build the terminal application which can be developed using the same hardware configuration as shown in fig.2.

b) Develop an additional program to the ROM operating system

The next level of sophistication in terms of a development strategy would be to add an additional application program which can be executed from the EEPROM memory. Clearly the ROM operating system must be designed from the start to allow this form of development. The COS (Card Operating System) supplied by Gemplus as a standard product allow for this enhanced application.

The COS operating system includes the concept of filters. This is a means by which the operating system can transfer control to an application program that has been loaded into the EEPROM memory. The developer will need to design his application using the appropriate software tools (an assembler is a minimum requirement) for the particular chip that is used by the IC card. Having developed this application then the machine code may be written into the EEPROM memory using the standard write memory command which is provided by the ROM operating system.

The trick provided by the COS operating system is the ability to set a vectored address in a defined memory location. When the operating system is executing it will examine this address at the appropriate moment (e.g on receipt of a command from the serial port) and if the address has been set then control will be transferred to the new application program residing in the EEPROM memory. This allows the additional application to manage the commands received over the serial interface. The COS operating system allows some flexibility in the way these filters are managed. From a developers point of view this approach is attractive because it allows a very fast development path. Security may be enhanced but without having to wait for the delay in reprogramming (and developing) the mask ROM operating system. The only draw back to this style of development relates to the use of valuable EEPROM space for storing the application program. The adequacy of the security features is a function of the operating system. There is no inherent reason as to why this should not result in a secure development path. It must of course be accepted that the developer of the operating system will be looking for a return on his investment but for many situations this must still lead to a cost effective development path.

c) Develop a new ROM operating system

This is of course the most involved of all the development paths. It is readily apparent that this can be a non trivial task and for an operating system with the complexity of the COS operating system referred to earlier almost certainly accounts for many man years of development. However for those situations where card cost is an overriding factor then the development of a ROM operating system allows the use of the EEPROM to be minimal or perhaps avoided all together.

The development environment in this situation is considerably more complex and results in the need for more components as shown in fig. 3. The centre of the development kit is the chip emulation system. This is manufactured to contain the components of the chip in an accessible form. Thus returning to fig. 1 the developer, by means of the development workstation is able to access the memory components separately in a form of test mode. Thus the workstation can load an operating system into the ROM memory area (usually implemented using RAM) and instruct the chip to execute the program. Like any normal ICE (In Circuit Emulation system) the developer can set break point and debug his program. The chip emulation system and in particular the probe is manufactured to behave exactly like the single chip equivalent. Some times this is achieved by using a bond out version of the IC card chip where all the address and data busses are made available to the emulation system.

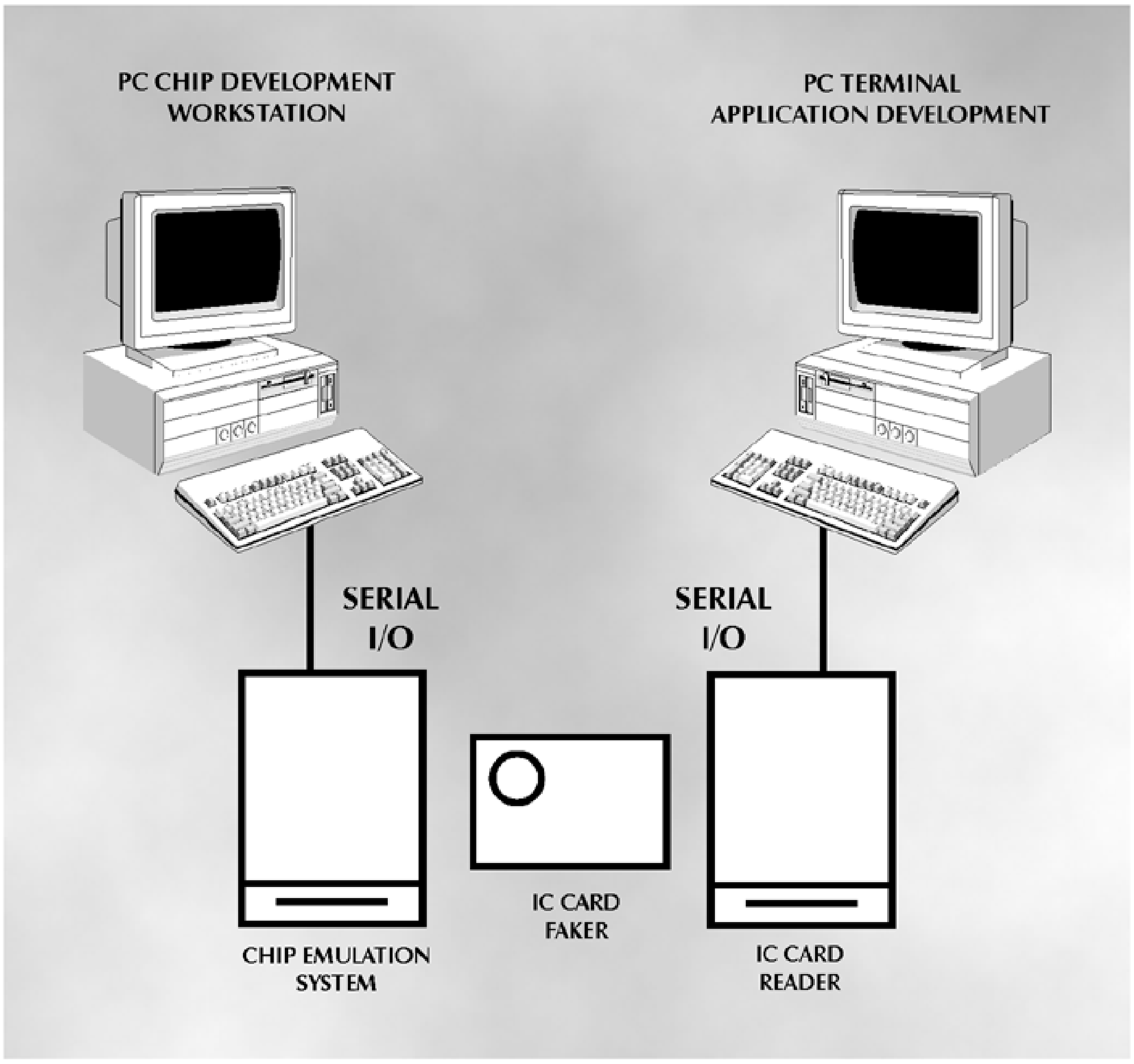


Figure 3: ICC Application Development Configuration

Having developed the new operating system in the emulator the total application can be tested by connecting a PC which contains the terminal end of the application. These two units are connected together by using what is sometimes called a faker card. That is a printed circuit board which has the same dimensions as an ISO 7816 -1/2 IC card at the connector end but which has a longer tail with a cable for connection to the emulator. The business end is plugged into a standard card reader (which can normally cope with the elongated tail) which is connected normally by a serial link to the PC terminal application.

By this means the developer can test the total application before having to manufacture the chip with the new operating system program. When the code has been tested as satisfactory in the emulation system then you can be pretty sure that when the chip is fabricated with the same code in the mask ROM that it will behave in the same way. This is not to suggest that the testing of the working sample produced by the semiconductor house can be avoided but only that one may reasonably expect success the first time round in the majority of cases where the initial emulation was fully evaluated.

David Everett

Next month: IC card security life cycle

Smart Card Tutorial - Part 12

First Published in August 1993

IC Card Security Life Cycle

The London Evening Standard reported this month of a new technique used in note counterfeiting of the US dollar. The method called "washing" refers to the concept of cleaning $1 notes with solvents to remove all traces of ink and then photocopying a $100 note onto the clean paper. Previously the task of creating paper with the right feel and appearance was considered far more difficult than the printing function. The modern colour laser photocopier has already become a serious counterfeiter's tool which is readily available at relatively low cost.

What is really apparent here is the problem of maintaining security throughout the life cycle of the $1 bill. The ability to work the paper for subsequent reprinting is clearly a major weakness. In our consideration of the Smart Card it is important to look at all stages in the life cycle of the card from conception to eventual destruction.

For the purpose of this part of the tutorial we will describe a hypothetical application for a finished Smart Card and will consider the complete life cycle and the appropriate steps to preserve the necessary security.

As we have mentioned previously security is a pervasive attribute and should encompass the complete application process. In any secure application there will be a mixture of security mechanism and procedural controls. Both are important for clearly the use of the strongest cryptographic mechanisms will be totally invalidated by insecure handling of the keys.

Let us consider a Smart Card with a single application that is used as a financial post payment card. This card will be used by the consumer to buy goods from the retailer. The card will check the users PIN and generate a certificate to authorise the transaction process. This will enable the retailer to receive funds from the card issuer who will correctly debit the customers account. Each of the stages in the card life cycle is shown in figure 1.

There are of course many variations for the IC card life cycle depending on the requirement of the particular application. In this example we are more concerned to show the principles rather than describe any particular scheme. Also for the purpose of this discussion we have simplified the cryptographic key management which we will describe in more detail in the next part of the tutorial.

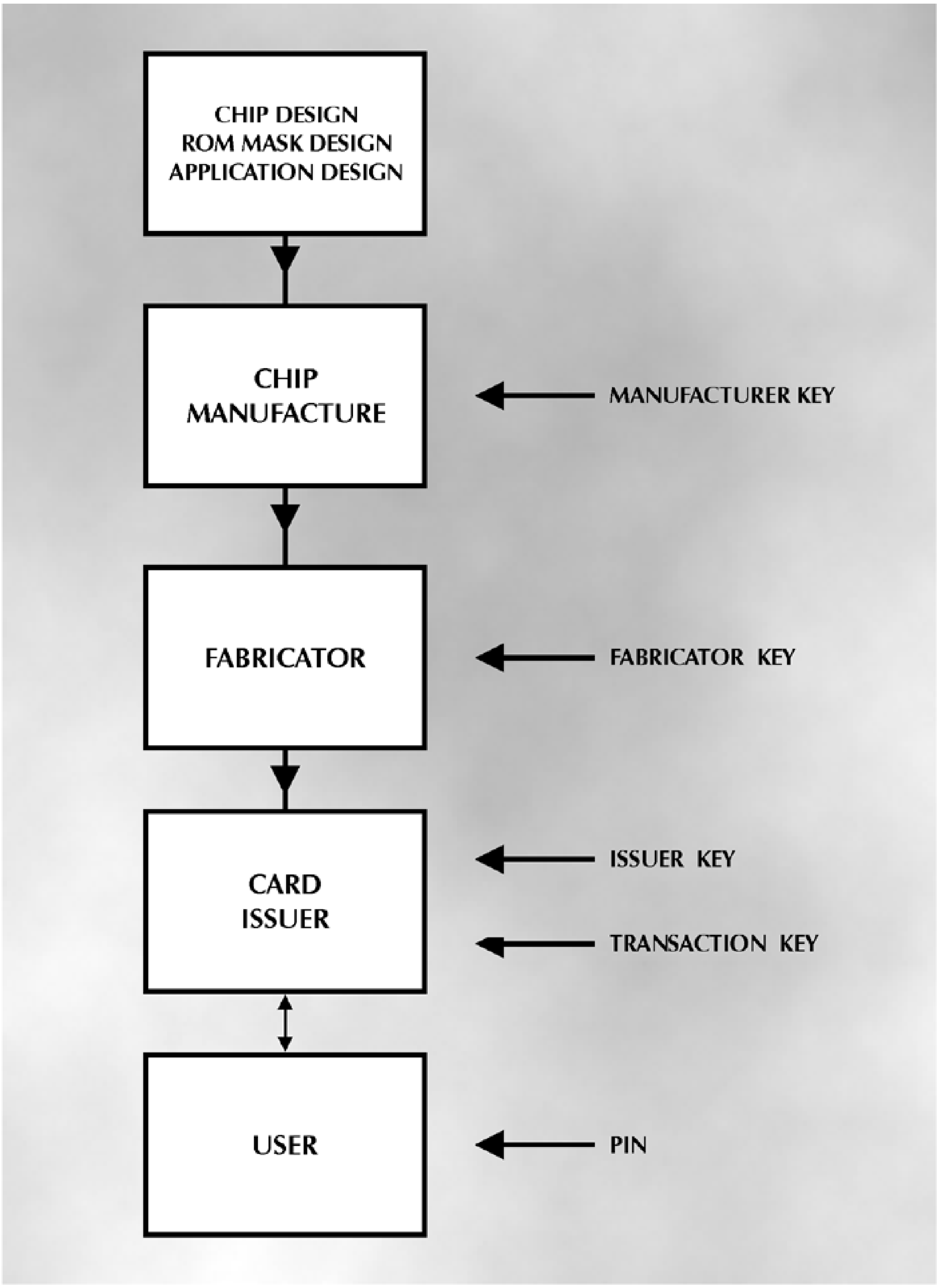


Figure 1: Financial Post Payment Card Life Cycle

In figure 1 we show a number of stages described by the role of an entity in the card life cycle. One of the key principles to be achieved is security segregation such that no one party can break the security controls.

The first stage in the process relates to the design of the operating system and the application. Although it would be possible to put the complete application in the ROM mask we will consider here two separate components. The operating system might even be general purpose but in any event it is clear that our life cycle must take this into account since it forms the base upon which the security of the application depends.

The chip design is of course fundamental to the overall security of the IC card. Much has been written about the physical security offered by a Smart Card. Anyone who doubts the sophistication of the technology required to manufacture integrated circuits would be advised to spend a day with one of the major semiconductor suppliers. You really cannot make a modern chip in your back bedroom. In fact you would find it very difficult to establish a semiconductor manufacturing operation without dealing with one or more of the small number of specialized equipment manufacturers. The environmental controls for such a facility are in themselves an extremely sophisticated operation where costs are calculated in the millions of dollars.

For the design and manufacture of the chips we are concerned with a complete material audit. Given the assumption that the attacker cannot make the chips then he will be obliged to steal them if he wishes to mount any form of counterfeit operation. The concept of "washing" will be referred to later. It is accordingly a requirement upon the chip manufacturer to account for all the chips that are made, some of which, due to yield failures, will need to be destroyed.

The design and development of the ROM mask operating system and the application software need to follow the usual principles for any software to be used in security applications. This is in itself a non trivial task but at least the memory available in Smart Card chips is relatively small which forms a limit on the eventual size of the software. The integrity and correctness of software code is a major subject in its own right but it is clear that the design methodology and subsequent testing must allow for both positive and negative testing. By positive testing we refer to checking that the functions defined in the specification are processed correctly. In the case of negative testing we are concerned to know whether the software does anything in addition to that defined in the functional specification. It is readily apparent that this negative testing is a difficult problem and in general cannot be guaranteed complete.

When the ROM mask software has been developed and tested the code is given to the chip manufacturer usually on an EPROM chip or on a floppy disk. He will then return an implementation of the code for cross checking before manufacturing the batch of chips. This is in itself a useful integrity check but clearly one normally requires this code to be kept confidential and therefore its distribution should be carefully controlled by the appropriate procedural measures.

The application software will normally be designed and developed by a separate path. The resultant code having been tested will be loaded into the PROM memory by a subsequent process. The chip manufacturer will produce a batch of chips containing the supplied ROM code (a test batch is normally produced for initial testing). The last part of the chip manufacturing process involves a test of the chip. At this stage the chip manufacturer would insert a secret manufacturing key into the EEPROM memory. The software in the ROM will have been designed to inhibit all functions without the correct presentation of this key. As such the chip is effectively locked.

The batch of chips is distributed to the fabricator whose task is to embed the chips into the plastic card. As we have discussed previously this involves a number of processes, where there will be some (nominal) failure rate. The role of the fabricator varies considerably between the various customers for their services. As a very minimum the fabricator must test the complete IC card to ensure its operational state. In some cases the fabricator completely personalises the card to the requirements of the issuer. For simplicity we will assume this latter position. In order to undertake this software identification and personalisation process the fabricator needs to `unlock' the chip by entering the manufacturer's key. As the last step in the personalisation process the fabricator will reset the manufacturer's key with a fabricator key before distribution to the card issuer.

The card issuer on receipt of the personalised cards will unlock the card using the fabricator key and will set the PIN for the user and the transaction key that will be used as part of the final application. The issuer will also reset the fabricator's secret key to the card issuer's secret key. The card is now enabled for operation and is distributed to the user.

The customer may use this card in a point of sale environment where the correct entry of the PIN is necessary before the Smart Card will generate an authentic transaction certificate. The retailer provides the transaction details (which will include the consumers account identifer) and the certificate to the issuer who will credit the retailers account and debit the customers account accordingly.

If the customer fails to enter his PIN correctly for a predefined number of trials then the application on the card will lock up. When the customer returns the card to the issuer then the application can be reset by means of the issuer key. Under normal operation the card should continue functioning until the expiry date set in the card data file is reached. At this stage the card will cease operation.

Now we can return to the `washing' concept. Can an attacker take the card at any stage and reprogram the data to his advantage. At each point in the life cycle the data on the card is protected by a secret key. Without knowledge of this key it is not possible to modify any of the card data in an unauthorised way. So here we have changed the attacker's work function to that of obtaining a security key held in the EEPROM memory compared with that of using chemicals to wash the $1 bill. I know which I would find easier to do!

Next month. Cryptography and key management.

David Everett

Smart Card Tutorial - Part 13

First Published in September 1993

Cryptography and key management

The particular advantage of a Smart Card with its inherent processing capability is the opportunity to implement cryptographic mechanisms within the Smart Card. As we have mentioned previously the IC chip may be considered as a tamper resistant module which offers significant resistance to physical attack. In this part of the tutorial we are going to take an overview of cryptographic algorithms and mechanisms along with their attendant key management considerations.

Although a large number of cryptographic algorithms have been developed over the years, in practice only two are in common use for financial applications which is still the main customer for such security. The DES (Data Encryption Standard) algorithm was proposed in 1977 and the RSA (Rivest Shamir and Adleman) in 1978. These algorithms represent two different classes of operation, DES is a symmetric algorithm whilst RSA is an asymmetric algorithm. The difference is easy to understand by referring to fig. 1. The input message is enciphered by means of key 1 to produce a cipher. The original plain text may be recovered by means of key 2. If both keys are the same (i.e Key 1 = Key 2) then the cryptographic process is symmetric. This is the more obvious operation and means that the sender and receiver of secret messages must share a common secret key.

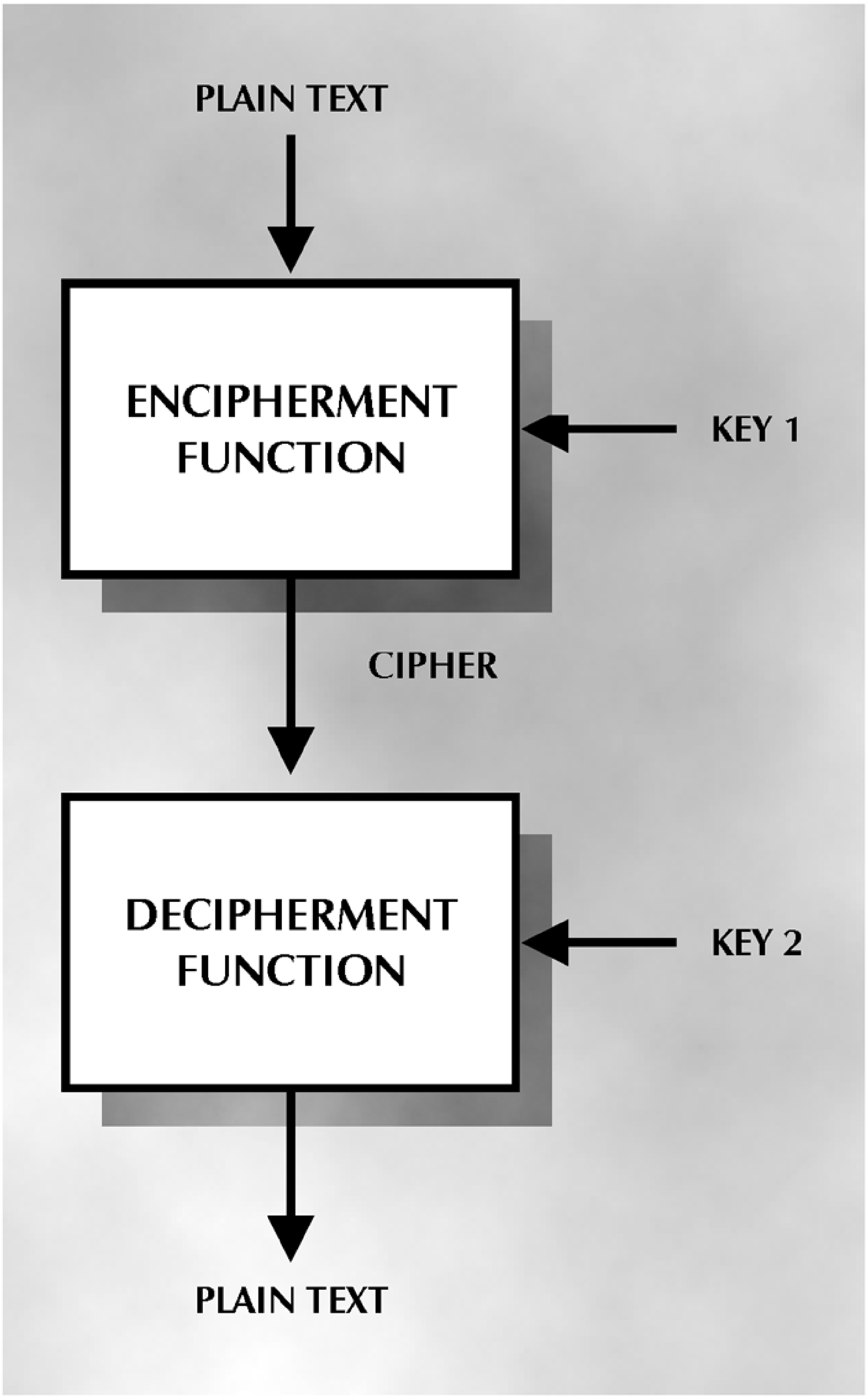


Figure 1: The Fundamental Cryptographic Process

The asymmetric situation is a relatively new concept first proposed by Diffie and Hellman in 1976 and represents the case where the two keys are different (but clearly related) and where it is not practically feasible to derive one key from a knowledge of the other. This form of asymmetric algorithm is often referred to as public key cryptography where it is assumed that one of the keys may be made public. Thus referring to fig. 1 it would be possible to make key 1 public for a particular entity. This means that anyone could produce a cipher using key 1 but only the owner of key 2 would be able to recover the original plain text. It should also be noted that the entity that creates the cipher using key 1 is equally incapable of reversing the process.

This concept of public key cryptography is an intellectual delight because at first sight it seems impossible. However it actually brings the whole concept of modern cryptography into perspective in as much that it shows the principle of the work function. These cryptographic algorithms are not absolutely secure but offer a resistance to attack defined by the relevant work function. In the case of the symmetric algorithm which uses a simple (secret) key, an attack could be based on trying all the possible keys until a sample of plain text and cipher text produce a direct match. The work function is then based on the time required on average to exhaust the total key space.

If we refer to the asymmetric case then we can show the concept of the work function by means of a simple analogy. Let us consider a message encoding and decoding system that is based on the use of dictionaries. The plain text message is represented by the English language whilst the cipher text may be represented by French (we conveniently ignore any similarity between some English and French words). The public key encoding process is implemented by giving all participants a dictionary that defines English to French only. This means that everyone can produce a cipher by turning an English message into French. However without the other side of the dictionary (French to English) there is a significant but not impossible work function to recover the original message. For each French word the whole dictionary would need to be scanned in order to find the English equivalent. In this simple analogy the intended receiver of these messages would be given the French to English dictionary. Clearly we can also build up a system by having other dictionaries (e.g English to German, English to Italian, etc) where all participants have the forward dictionary but only one has the reverse dictionary.

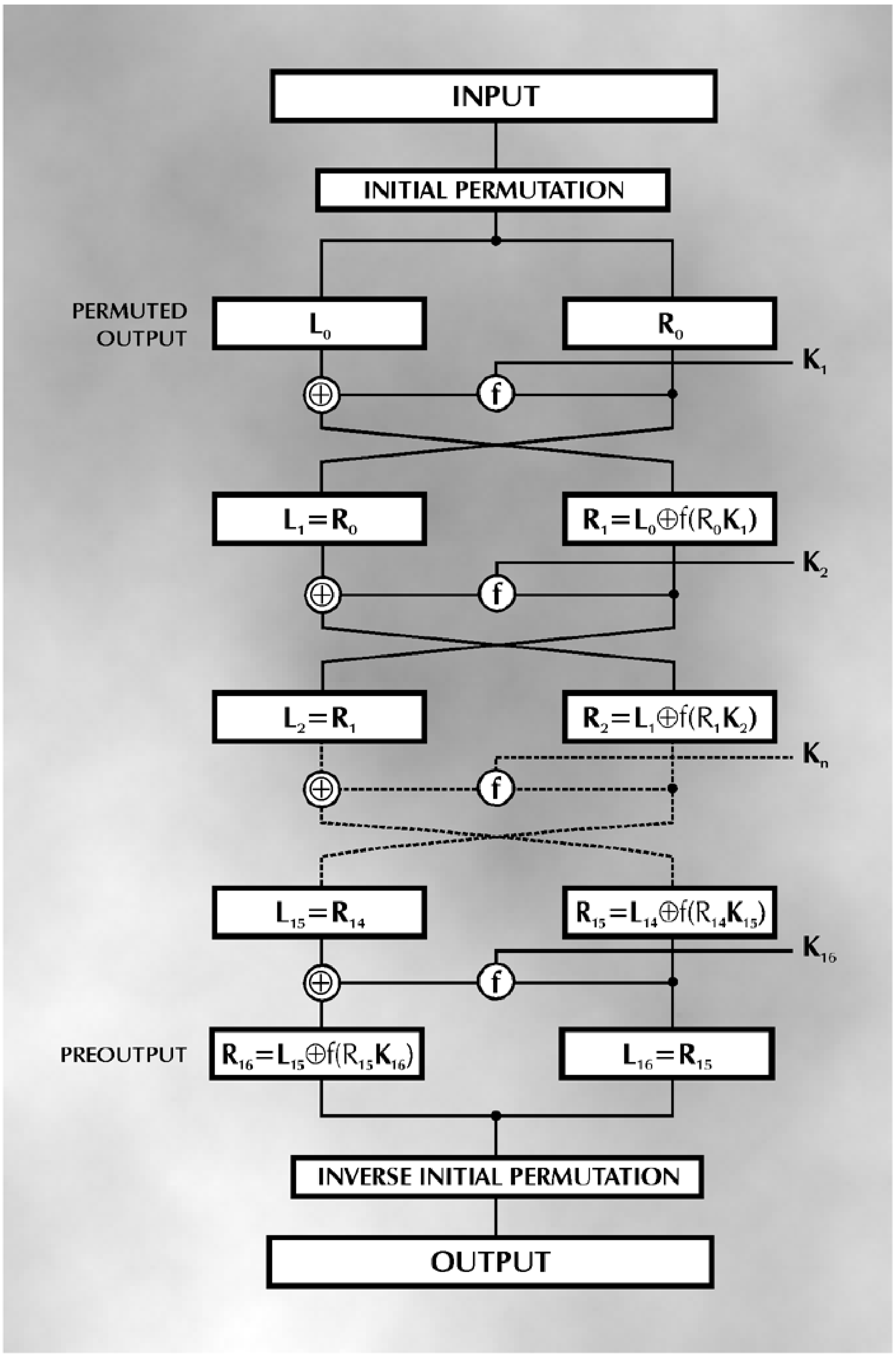

INPUT
INITIAL PERMUTATION
PERMUTED OUTPUT
L_0
R_0
K_1
f
$L_1=R_0$
$R_1=L_0\oplus f(R_0K_1)$
K_2
$L_2=R_1$
$R_2=L_1\oplus f(R_1K_2)$
K_n
$L_{15}=R_{14}$
$R_{15}=L_{14}\oplus f(R_{14}K_{15})$
K_{16}
PREOUTPUT
$R_{16}=L_{15}\oplus f(R_{15}K_{16})$
$L_{16}=R_{15}$
INVERSE INITIAL PERMUTATION
OUTPUT

Figure 2: The DES Algorithm

This conveniently leads us to a further concept of the one way function (OWF). This is very important in modern cryptography and underlies the use of many modern mechanisms. The principle is very straightforward and may be considered analogous to the lobster pot. The design of the pot is such that it is very easy for the lobster to enter the pot through the entry tube but the reverse process is practically infeasible (to even the most athletic lobsters). It is the next stage that is also important, the fisherman recovers the lobster by means of a trap door in the bottom of the cage for which only he has the key. This is just another way of looking at the public key algorithm as a one way function with a trap door. In all cases of course we can show an appropriate mathematical representation. The DES algorithm is shown in fig. 2.

The DES Algorithm

The DES algorithm was initially published as FIPS publication 46 (USA Federal Information Processing Standards) in 1977. The algorithm is designed to encipher and decipher 64 bit blocks of data using a 56 bit key. The process is shown in fig.3. The block to be enciphered is subjected to an initial permutation (IP). The output of this operation is then iterated 16 times by the following operation,

- $L' = R$
- $R' = L \oplus f(R,K)$

Where L = left most 32 bits of previous step
- R = right most 32 bits of previous step
- f(R,K) = function of key and R

at the end of this loop the result is put through a final inverse permutation (IP^{-1}) to produce the 64 bit output.

The decipherment process operates in the same way except the key function is used in reverse order.

The RSA Algorithm

The RSA algorithm has an attractive elegance about it, probably because of its apparent simplicity as shown below,

- $C = M^e \text{ Mod } N$

- $M = C^d \text{ Mod } N$

Where:

- M = Message Block; C = Cipher Block
- e = Encipherment key; d = Decipherment key
- N = Modulus (product of two primes p and q)

and

- $de = 1 \text{ Mod } \operatorname{lcm}(p-1, q-1)$

Like all cryptographic algorithms there is much under the surface and there are a number of conditions that must be met in order to implement the algorithm correctly. This need not concern us here but we can look at an example using trivial numbers just to see the algorithm in operation,

- let M = 4
  p,q = 5,11
  N = 55 (p*q)
- choose e = 3
  d = 7 (de = 1 mod 20)

Encipherment

- $C = 4^3 \bmod 55 = 9$

Decipherment

- $M = 9^7 \bmod 55 = 4$

(note: $9^3 \bmod 55 = 14$)

In practice the size of N and d are typically 2^{512} which is about 154 decimal digits. These numbers are difficult to process quickly with the 8 bit CPU's commonly found in the Smart Card microcontroller. We will return to this subject in more detail later in the tutorial.

Security Mechanisms

There are a number of security mechanisms that can be used in Smart Card applications but of particular interest are those mechanisms that relate to data integrity and authentication. The cryptographic check values (CCV) and digital signatures are the most widely used mechanisms, sometimes the term signature is applied to both mechanisms.

Cryptographic Check Values(CCV)

The use of the DES algorithm to calculate CCV's is well established. The often used MAC (Message Authentication Code) was originally defined by the ANSI X9.9 standard and subsequently adopted as the ISO 8730 standard for finacial messages.

This check value is generated by using the DES algorithm in cipher block chain mode as shown in fig.3. The standards referred to above use the most significant 32 bits of the final output as the check value or MAC. If necessary the final message block is padded out with zeros, so that each input block is always 64 bits. The receiver checks the cryptographic check function by applying exactly the same operation.

The primary purpose of the CCV is to provide a data integrity function that ensures that the message has not been manipulated in any way. This includes both the modification, addition and deletion of data. In itself this function does not provide any such assurances on the addition or detection of whole messages. The necessary security can be acheived however by the use of sequence numbers which are incorporated within the CCV.

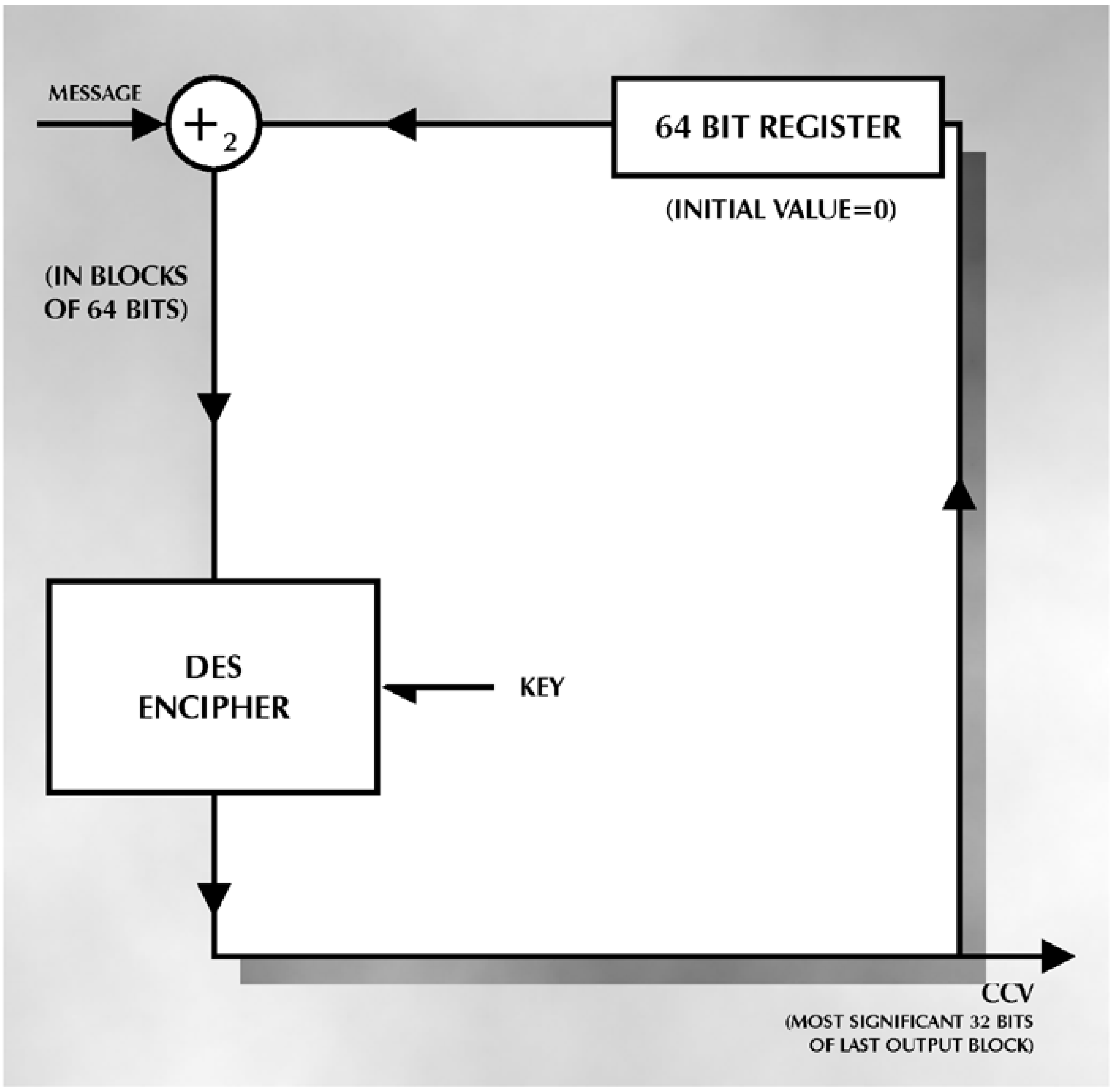


Figure 3: The Cryptographic Check Value

The cryptographic check value also supplies some assurance of source authentication depending on the key management architecture. As a trivial example where two correspondents (only) share the secret DES key then the receiver can be assured of the authentication of the sender. The CCV does not however provide the property of non - repudiation. In the example shown here the receiver has the same secret key as the sender of the message and clearly is capable of modifing or inserting a new message. Under such an arrangement it is difficult to prove the source authenticity of the message to a third party. This is better provided by the use of an asymmetric cryptographic algorithm.

David Everett

Next month. Part 14 - Cryptography and key management continued.

Smart Card Tutorial - Part 14

First Published in October 1993

Cryptography and key management (continued)

Digital Signatures

The availability of public key cryptography algorithms has led to the adoption of a range of digital signature mechanisms. These signatures not only produce the properties of data integrity and source authentication but also effectively meet the requirements for non - repudiation. A digital signature may be generated by means of the RSA algorithm as shown in fig.1.

The message is reduced to a digest by means of an appropriate hashing algorithm. The resultant digest is calculated to be smaller than the block size of the RSA algorithm (typically 512 bits). This digest is then processed by the RSA algorithm as shown below using the secret key of the sender. The receiver takes the signature and applies the algorithm using the public key of the sender. The receiver also processes the message to calculate the message digest and compares the result. This form of signature is sometimes referred to as a signature with appendix because the message needs to be sent along with the signature. If the message is smaller than the block size of the algorithm then the hash function could be omitted to produce an impressed signature. In this case it would not be necessary to send the message since this data would be recovered by the RSA encipherment process. It should however be noted that there is a fundamental requirement for any signature process to incorporate adequate redundancy (say 128 bits). This means that a necessary amount of deterministic data must be included in the input to the signature creation process.

Signature generation

- $S = M^d \text{ Mod } N$ (equivalent to the decipherment operation)

Signature checking

- $M = S^e \text{ Mod } N$ (equivalent to the encipherment operation)

- Where M = Message block (or digest)
- S = Signature
- e = Public key (of sender)
- d = Secret key (of sender)
- N = Modulus

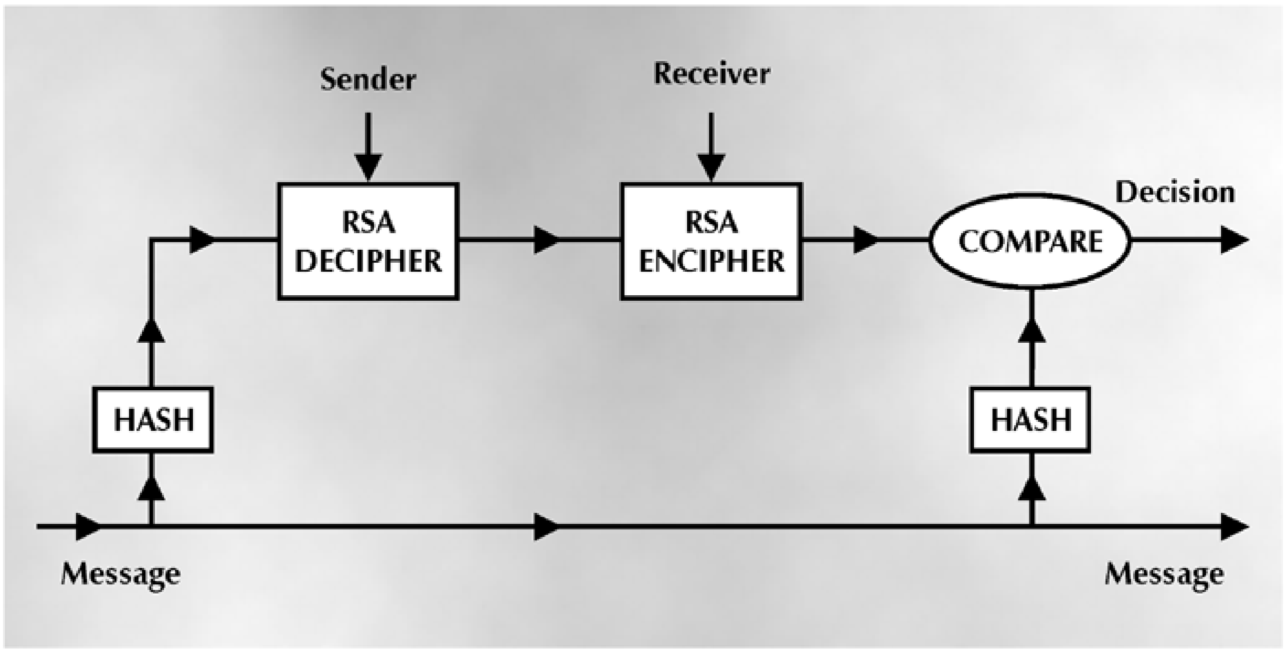


Figure 1: A RSA Digital Signature

The particular points to be noticed about the use of the RSA algorithm are that the block size is set by the choice of modulus (product of two primes) and that the encipherment key may be chosen to be very small (often e = 3). The modulus N is common to both the signature generation and checking process and the size of the secret key d will be the same size as N. It is now readily apparent that the signature creation process is much slower than the signature check operation being the ratio of 576 2 modular multiplications on average.

There are two obvious vulnerabilities with digital signature algorithms that must be addressed in the design of a secure system. In the first instance it is clear that it is very easy to generate an apparently authentic copy of a digital signature since it has none of the properties necessary for forensic evidence that may be applied to a written signature. The second problem relates to the authenticity of the keys. Here we have shown a process where the sender supplies both the keys for signature generation and checking. It is clearly essential to use some additional process to be assured of the authenticity of the senders public key. We will look at this in more detail when we discuss key management. Without this proof of authenticity neither of the properties of source authentication or non - repudiation can be substantiated.

The digital signature algorithm (DSA)

This relatively new algorithm was first proposed by NIST (U.S. National Institute of Standards and Technology) in 1991. It is different from RSA in that it is designed only for the creation and generation of digital signatures and not for the encipherment and decipherment of data as may be achieved by the RSA algorithm. The DSA algorithm is defined as follows:

Global constants

- p = a 512 bit prime number
- q = a 160 bit divisor of p - 1
- g is chosen such that $g^q = 1 \bmod p$
- ($g = a^{p-1/q} \bmod p$)
- where $0 < a < p$

For each entity

- Choose a secret key x $0 < x < q$

- Compute a public key $y = g^{x} \bmod p$

Signature generation

- apply a secure hash function to the message to calculate H.

- Compute:
- k = A random number $0 < k < q$
- $r = (g^{k} \bmod p) \bmod q$

- $s = (K^{-1} (H + x r)) \bmod q$

Signature verification

- Compute:

- $t = s^{-1} \bmod q$

- Check:
- $r = ((g^{Ht} y^{rt}) \bmod p) \bmod q$

One cannot resist wondering which is the better signature algorithm? Before we can show some comparisons it is appropriate to look just a little further at an identity verification algorithm invented by Fiat and Shamir.

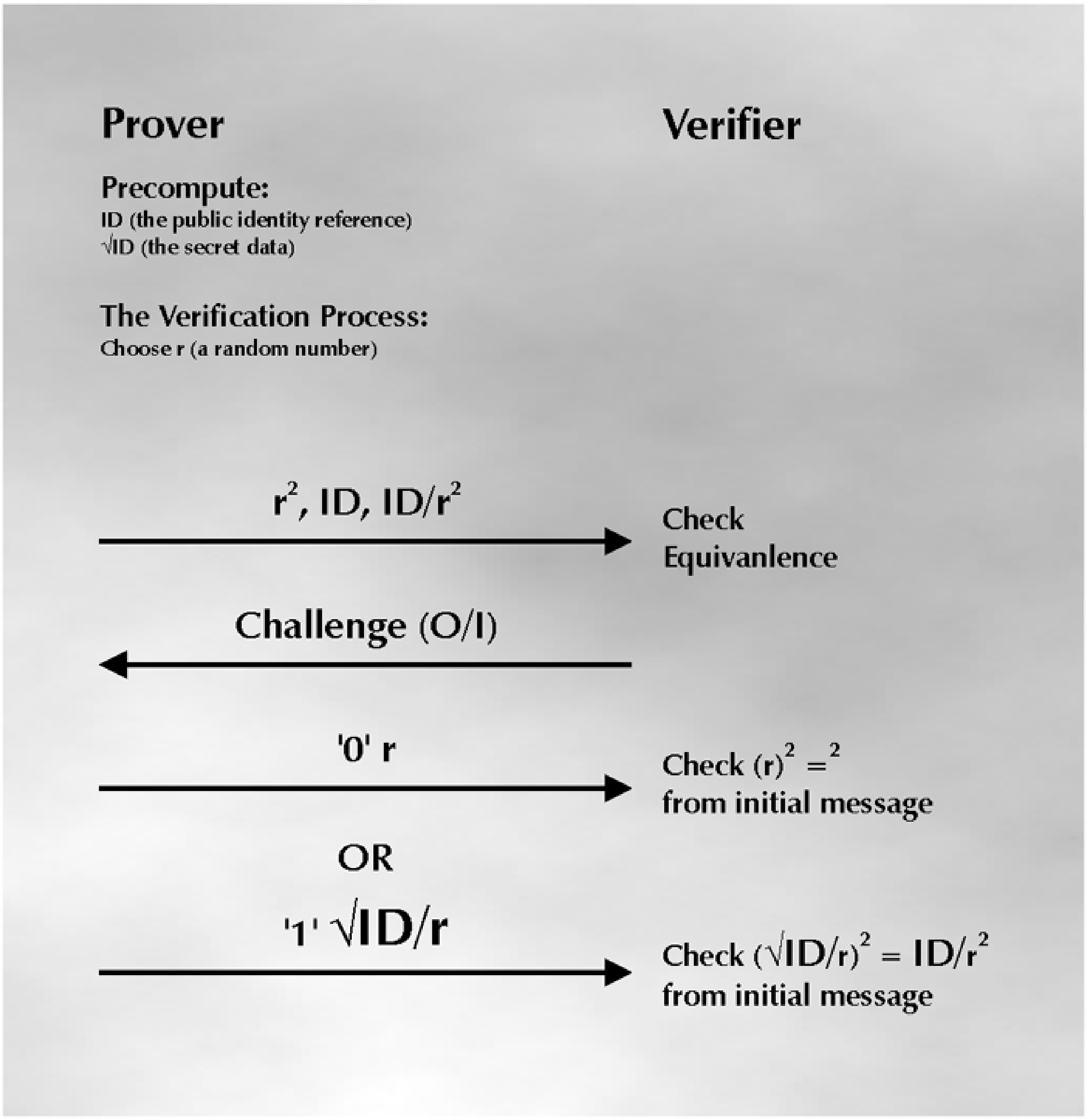
The Fiat - Shamir Identity Algorithm

This algorithm represents a class of zero knowledge proofs where it is possible to show that you know a secret without actually revealing the secret. The trick to this method relies on the difficulty of computing a square root mod m without knowledge of the constituent primes. A reference number ID is chosen by the central key authority as a quadratic residue so that it can calculate √ID mod m as a pre-computation. This √ID represents the secret.

The algorithm uses a global modulus m which is the product of two primes (known to the central key authority) as was shown for the RSA algorithm. The process takes place as shown below where all computations are assumed to be mod m.

It is readily apparent that if the prover supplies both responses for a particular r then the secret would be revealed since,

- $r. \sqrt{ID}/r = \sqrt{ID}$



Just one invocation of the challenge process is not sufficient since the prover may be lucky to get away with just proving the square root of the random number r^2. However if the process is repeated twenty times then the probability of a masquerade is 2^{-20} or about one in a million.

This identity algorithm has been extended by Fiat and Shamir to act as a digital signature algorithm. The challenge is based on the message to be signed and vectors are used for ID and √ID to develop a sufficient signature size.

It is now possible to make some comparison between the different signature algorithms in terms of the size of the data elements and the number of modular multiplications as shown in the table. These are of course the important parameters when operating with Smart Card microcontrollers.

	RSA	Fiat - Shamir	DSA
Public key size (bytes)	64	576[1]	212[2]
Secret key size (bytes)	64	576	20
Signature generation: No of squares	511	8	159
Signature generation: No of multiplications	255	36	79
Signature checking: No of squares	1[3]	8	159
Signature checking: No of multiplications	1	36	119
Signature size (bytes)	64	521[4]	40

Notes:
1) Can be reduced by using an algorithm for computing from a public ID
2) Includes the common data p, q and g
3) This assumes an encipherment key e = 3
4) $\Sigma\, y^i = 512$; $\Sigma\, e^{ij} = 9$ (Fiat Shamir typical signature parameters)

David Everett

Next month. Part 15 - to be continued.

Smart Card Tutorial - Part 15

First Published in November 1993

Cryptography and key management (continued)

We have discussed a range of cryptographic algorithms for confidentiality, authentication and data integrity security services. We have also implicitly assumed that these algorithms are adequately secure in that their public knowledge will not invalidate the effective security. In other words the security of the particular security service depends on the management of the cryptographic keys. Whilst this is an achievable principle it is readily apparent that knowledge of the particular algorithm used does in fact give the potential attacker valuable information. In the military world the algorithms are kept secret to ensure the security and it is only the practicabilities of the situation that result in the more open use of the algorithm for commercial use. The use of Smart Cards is an advantage here since it is operationally possible to distribute IC cards to all participants without revealing the contents. The tamper resistant properties of the IC card make this an ideal carrier for both cryptographic keys and algorithms.

The IC card as a tamper resistant module.

The concept of tamper resistance is well established in the world of cryptographic security equipment. We have deliberately avoided the use of the word tamper proof as this is technically unachievable. As a starting principle it is a reasonable concept to compare the properties of tamper resistance with that of the security of a cryptographic algorithm. In both cases we are dealing with the work function required for a successful attack. In the case of the cryptographic algorithm we can consider the resources required to achieve a brute force attack by key exhaustion or in the case of the RSA algorithm for the factorisation of the modulus. We must also allow for some logical flaw in the algorithm that may result in some short cut for a successful attack. It is the latter point that is non deterministic.Since it is never possible to prove the absolute security of a cryptographic algorithm. In this situation our confidence in the strength of the algorithm depends on its exposure to expert analysis. Both the DES and RSA algorithms have withstood this public exposure to date resulting in some interesting methods of attack but that none the less still leave the algorithms with secure pedigrees from a practical point of view.

Tamper resistance is equally interesting and has occupied the minds of designers for a number of years. If we accept that we are just playing with the work function required to achieve a successful attack then it is easier to accept that adequate security can be provided. In a commercial world we normally define this adequacy in terms of a work function that ensures an attack is not economically viable. It is clear that this work function then depends on three parameters,

- time
- skill level
- resource availability

Time is fundamental to any security scheme and it is the primary task of the designer to ensure that the time required to make an attack, either exceeds the lifetime of the asset being protected or is such that the cost of pursuing the attack offers insufficient benefits. There should be no doubt in the readers mind that professional criminals are well equipped to develop the business case for their activities.

Skill levels should not be underestimated as an important factor in determining the security work function. In particular this parameter is closely linked to the rapid development of the relevant technology in this area. A quick look into the back of a modern television set compared with 20 years ago gives some dimension to these advances. The modern microcontroller used for Smart Cards is a brilliant feat of engineering that seems to know no bounds.

The last parameter refers to the need to acquire the necessary resources to effect an attack. When considering cryptographic algorithms we immediately turn our attention to the availability of the necessary computer resources and their effective MIPS (Millions of Instructions Per Second). Of course in this case we also need to take account of the increasing power of these machines and the now wide scale use of networks that enable groups of machines to be harnessed to attack a single problem. In the case of the IC chip we are concerned with more specialist equipment and their somewhat higher price tags. Modern silicon wafer fabrication lines are priced in billions of dollars. Furthermore the increasing complexities of the technology is such that these prices are unlikely to reduce.

We can define a tamper resistant device as one that offers both physical and logical protection against unauthorized access to its secret data. In our case that will be at least the cryptographic keys and perhaps the algorithm and other more general security data.

In terms of physical attacks the tamper resistant device should form a barrier to an invasive attack. There may be a number of barriers either physically hard or deliberately brittle such that an attack will evoke a response to eliminate the secret data. The classical bank safe forms a hard barrier to an attack but may also include invasion sensors that sound appropriate alarms when invoked in an unauthorized fashion. An integrated circuit chip may be encapsulated in such a way that removal of the barrier either damages the device or triggers sensors that may be used to eradicate the secret data. Some chips for example incorporate sensors in the passivation layer that set flags in a security register that may be interrogated by the application software.

It should be noted here that there is a difference between reverse engineering an IC chip and obtaining the data contents. As we have mentioned previously the data is stored in the ROM or EEPROM memory. In the case of ion implanted ROM and the EEPROM there is no obvious way of obtaining a visual image of the data contents. Thus the technique of reverse engineering, which are themselves extremely specialist, may result in the production of an electronic circuit diagram but do not reveal the data contents of the memory.

The subject of chip security is an important issue and one that we will refer to again but suffice it to say at this stage that the modern IC chip can form an extremely effective barrier to an invasive attack. Whilst some chips are clearly better protected than others the modern advances in technology are significantly on the side of the security designer. In terms of logical security this is very much in the hands of the designer and clearly steps should be taken to ensure that adequate security is achieved. For the moment it is reasonable to suggest that the IC chip can offer a tamper resistant module with a security work function adequate to meet the emerging commercial needs.

Key Management

This is really the crux of cryptographic security and it is in this area that the differences between symmetric and asymmetric cryptography become most obvious. Let us first look at symmetric cryptography as might be experienced using the DES algorithm. The whole purpose of security is to effect a security service between two or more entities. It is therefore readily apparent that a common key must be established between these entities before the security service can be effected. Let us consider the practical situation of an IC card effecting an application with a terminal. In fig.1 we show a simple situation where a common key (k) has previously been established in both the IC card and the terminal.

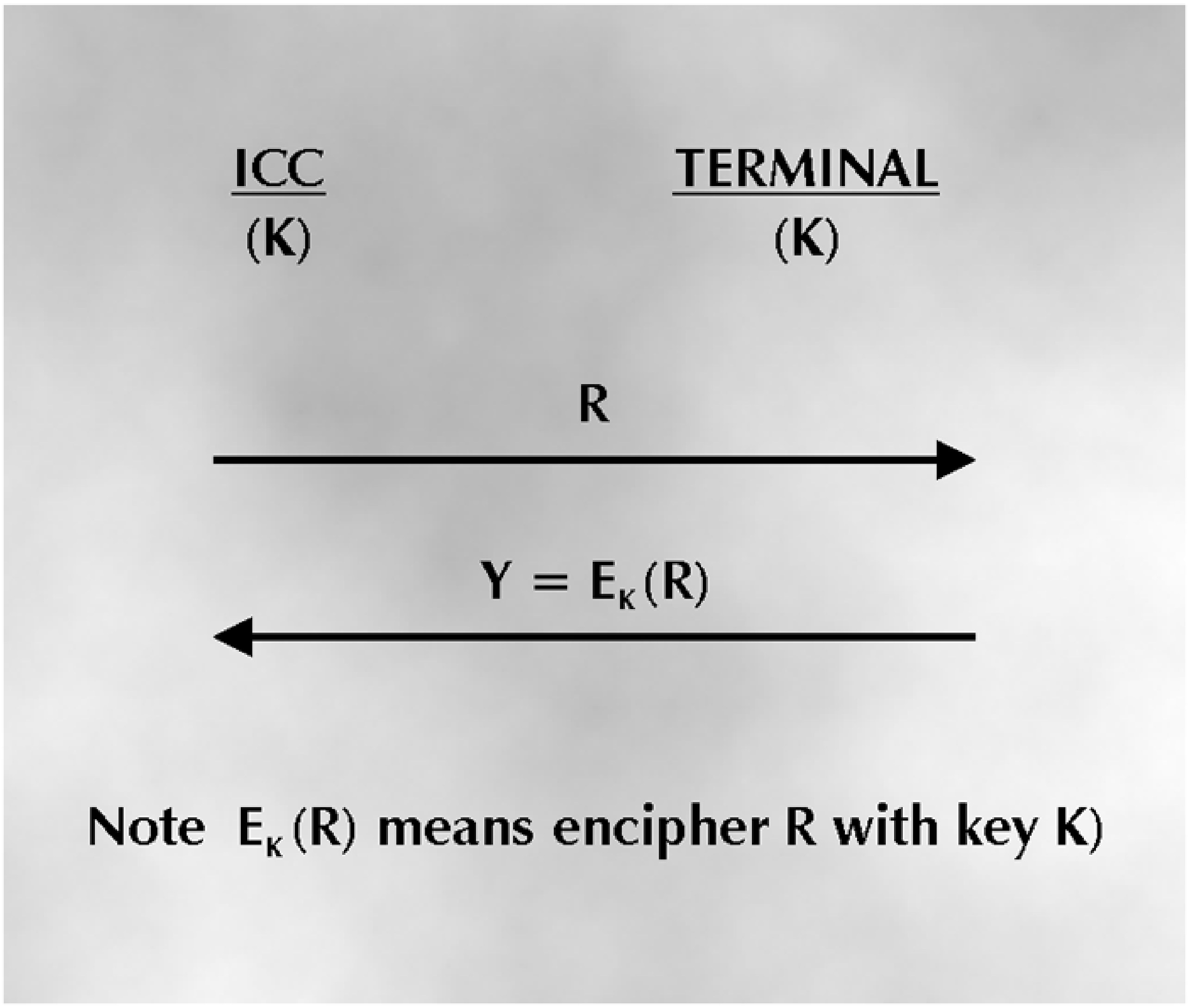


Figure 1: A Simple Common Key

As an example we shall consider an authentication process where the terminal checks the authenticity of the ICC. Here the terminal sends the ICC a random number R. The ICC enciphers this number with the common key K and returns the response Y to the terminal. By using the same algorithm and key the terminal can check that the ICC knows the algorithm and common key K. It is readily apparent that in this scenario we would need to establish a global secret key in all the ICC's and terminals. Not only is this operationally difficult but any breach in security in any card or terminal would expose the global key. Clearly the tamper resistant properties of the terminal should be no less than that of the ICC.

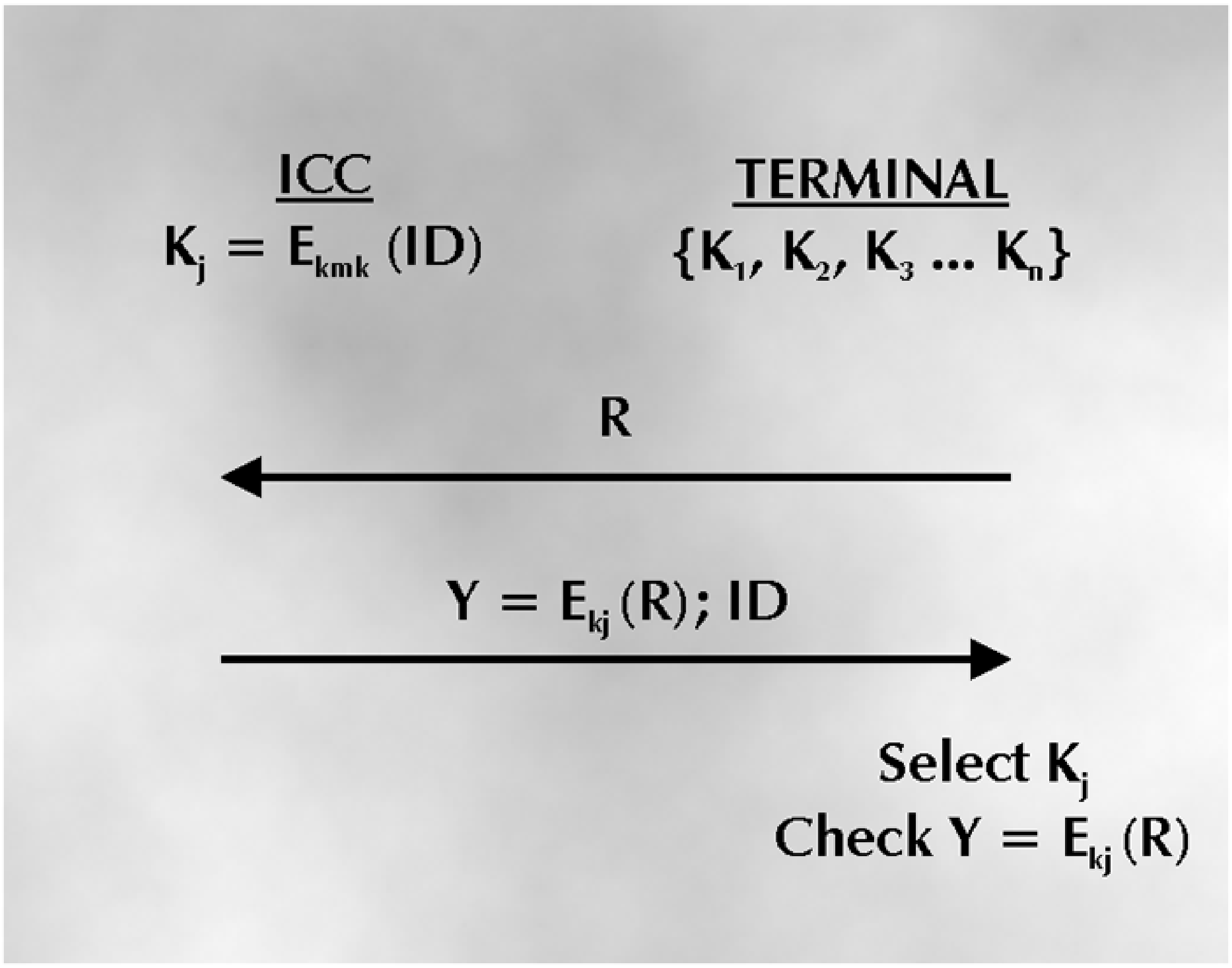


Figure 2: Terminal Key Sets

We can improve our security vulnerability by using sets of keys in the terminal to achieve a level of security segregation as shown in fig 2.

In this situation a breach of security in the ICC only reveals the key Kj. However an attack on the terminal would reveal the complete set of keys. The assumption here is that the security of the terminal is higher than for the ICC.

Another variant of the key management process is to use derived keys as shown in fig.3

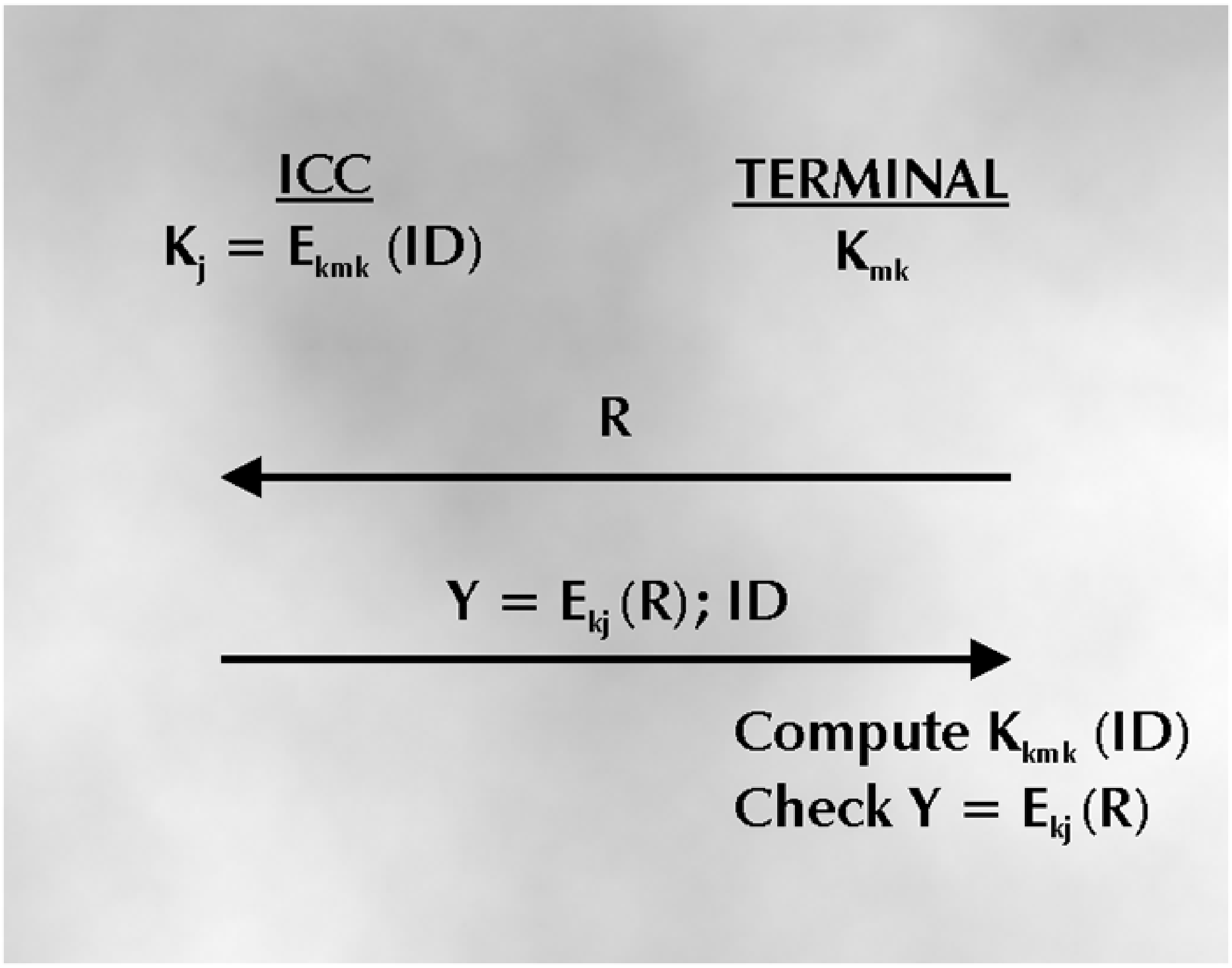


Figure 3: A Derived Key

Here the terminal contains a master key K^{mk} whilst the ICC has been preloaded with a key derived from this master key. For example this might be an enciphered form of its identity ID.

In this scenario the global security depends again on the terminal but each ICC can have a unique derived key. This means that the effect of an attack on an individual ICC can be restricted to that individual card. In the previous case the segregation is proportionable to the size of the key set. In other scenarios it is possible also to have sets of master keys to generate a particular unique derived key.

Clearly the use of a unique key per card is a powerful security advantage but it must be appreciated that the terminal master key exposes the security of the system as for the first scenario.

David Everett

Next month, Key management continued.

Smart Card Tutorial - part 16

First Published in December 1993

Cryptography and key management (continued)

The difference between symmetric and asymmetric cryptographic algorithms is most obvious when examining the various key management architectures of the two types of algorithms. This month we will build up a typical asymmetric structure in a similar fashion to that used previously for the symmetric case.

We have frequently intermixed the terms asymmetry and public key in order to maintain correspondence with the term used in the general literature. The concept of a matching public key (PK) and secret key (SK) will be used constantly in this discussion but clearly the actual use of the public key will be decided by the designer or a particular security system. Such keys may well be publicly available whilst in other situations their confidentiality may be maintained as an additional security feature. In the case of RSA this public key relates to the modulus since the public exponent is normally chosen as a global constant. This exponent is often chosen to take the value of 65,537 (2^{16}+1); Fermat's number F^4) or 3 (F^0; (2^1+1)

In both cases the numbers are chosen as particular primes which results in a small number of multiplication when calculating the exponent. The value of 3 is obvious whilst 65,537 in hex is 10001. Every 1 bit in the exponent adds an additional multiplication when calculating the exponential. Although the discussion that follows here assumes the RSA algorithm for calculating digital signatures the arguments presented are equally applicable to other signatures schemes such as the DSS (Digital Signature Standard).

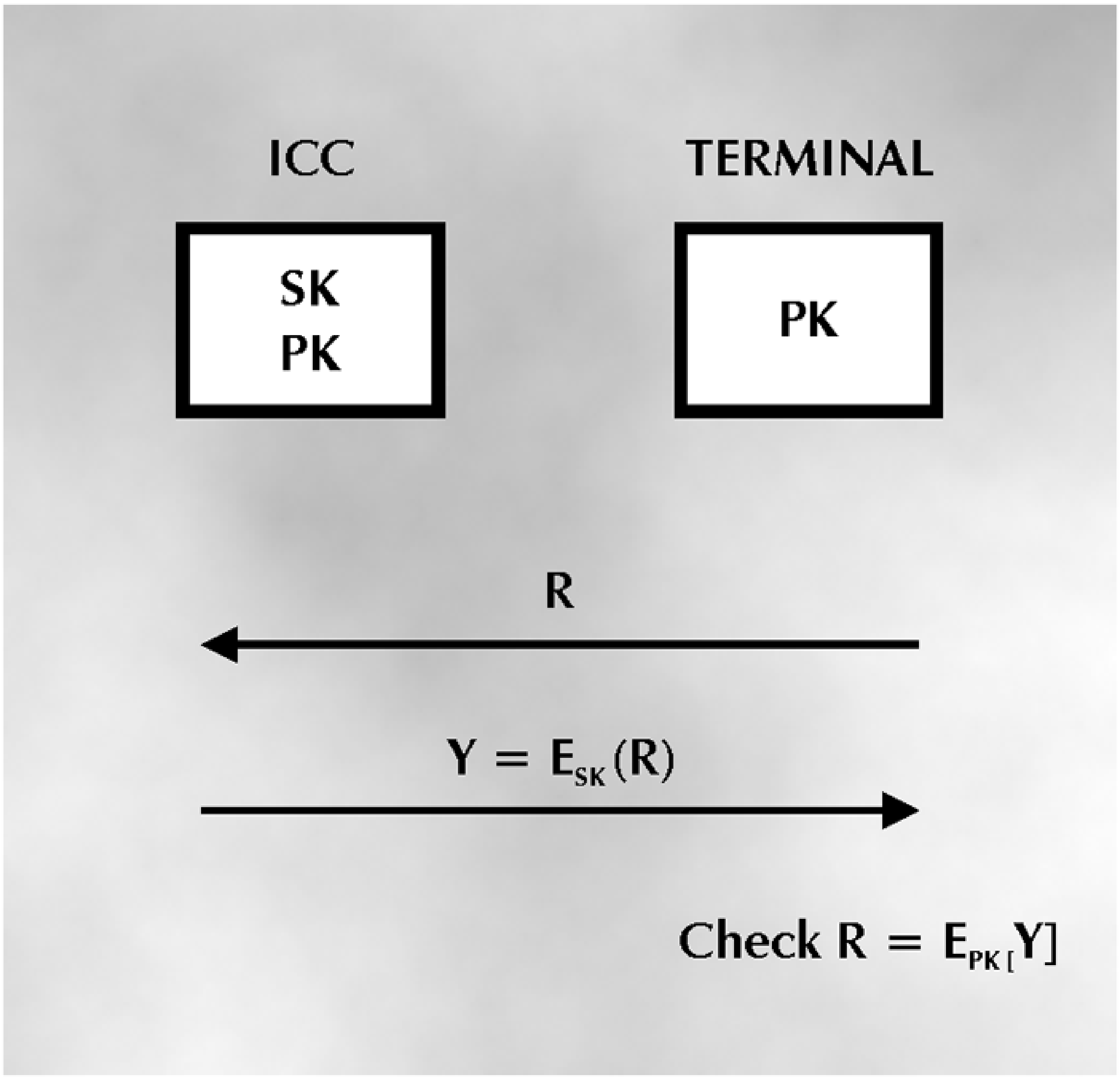


Figure 1: A Simple Asymemetric Authentication Process

Let us first look at the single authentication method presented last month but this time using the RSA algorithm as shown in fig.1. Here the terminal sends the ICC a random number R. We should note also that the point of using a random number is only meant to portray data that cannot be predicted by an attacker or more to the point represents data for which he is unable to precompute the response. In practice the terminal may present a number of data fields including perhaps the current time. The ICC enciphers the random number R using its secret key SK as the exponentiator (R^{sk}Mod N). The result of this computation Y is returned to the terminal which then uses the matching PK to check for correspondence with the supplied R (i.e check if R = Y^{pk}Mod N).

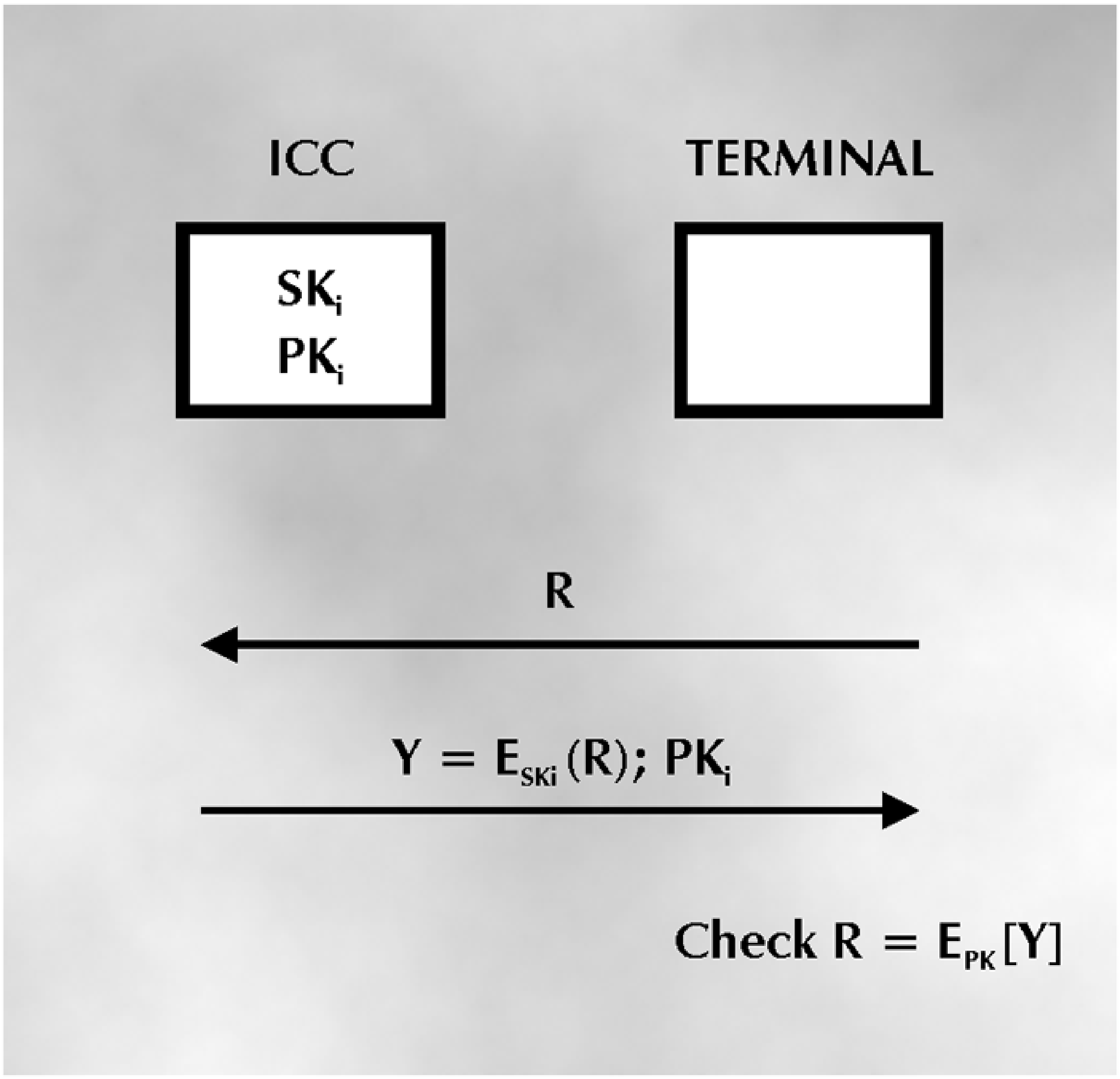


Figure 2: Unique Key per Card

In this simple example we have assumed that all the ICC's contain a common secret key whilst all the terminals store the matching common global public key. Clearly the system is exposed if one can determine the secret key of any ICC. An attack on the terminal to expose the public key has no value although the ability to change the key allows an attack that we will discuss later.

We can improve the security segregation by having a unique key in each ICC. In this situation (fig. 2) it is necessary for the terminal to know the matching public key. In the scenario shown in the figure the ICC could of course just present its public key to the terminal for checking the signature. There is of course a snag and it relates to the authenticity of the public key presented by the ICC. If no checks are made on the genuineness of this public key then an attacker could make up his own secret key and public key pair and would be guaranteed to pass the test.

This leads us towards the basis of all cryptographic systems, the centre of trust. If two unknown parties wish to correspond then there needs to be a common point that they are both prepared to trust. The role of this trusted entity is to either supply cryptographic keys or to vouch for the authenticity of a key generated by the particular entity. This centre of trust is often referred to as a global key centre (GKC) because its primary role is concerned with the key management of the particular security system.

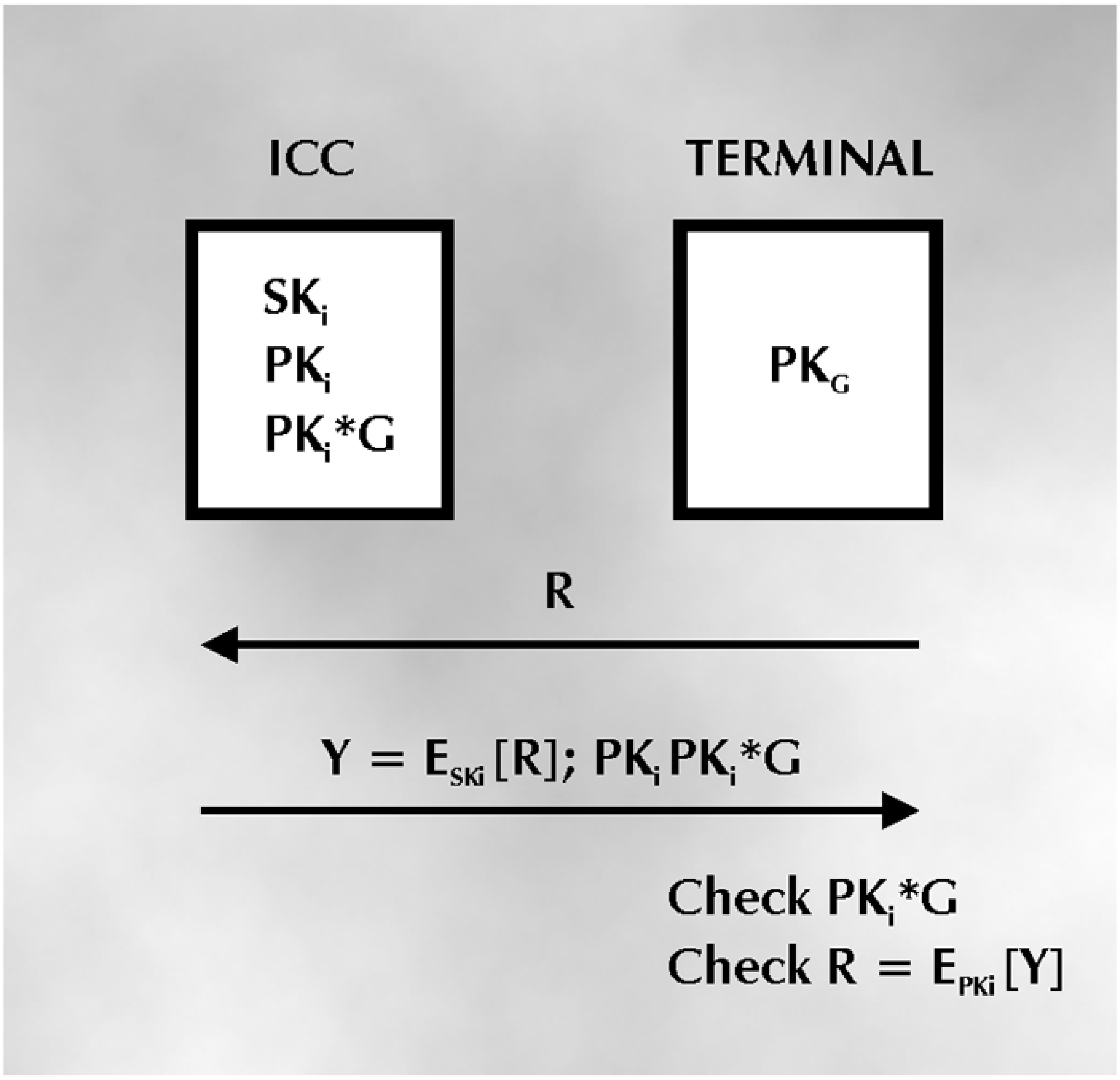


Figure 3: Public Key Authentication

In a public key system this GKC has its own secret key and public key pair. It uses the secret key to produce a signature for the public keys of the participating entities. This signature is often referred to as a key certificate. We have used PK^{*G} as a representation of a key certificate on PK produced by the entity G. It is now only necessary to distribute the public key of the global key centre (PK_G) to all the terminals. In fig.3 we show how this can operate. The ICC now sends to the terminal its public key PK_i and the certificate (produced by the GKC) for this key $PK*i^G$. In checking the response to the random number challenge the terminal carries out two operations. In the first instance it checks the authenticity of the public key presented by the ICC using the public key of the GKC (checks $PK_i=E_{PKG}[PK*_i^G]$).

Having checked the authenticity of the ICC's public key the terminal can then check the response to the random number challenge using the public key of the ICC. It should be noted that the ICC needs to store both its own public key and secret key and additionally the key certificate supplied by the GKC.

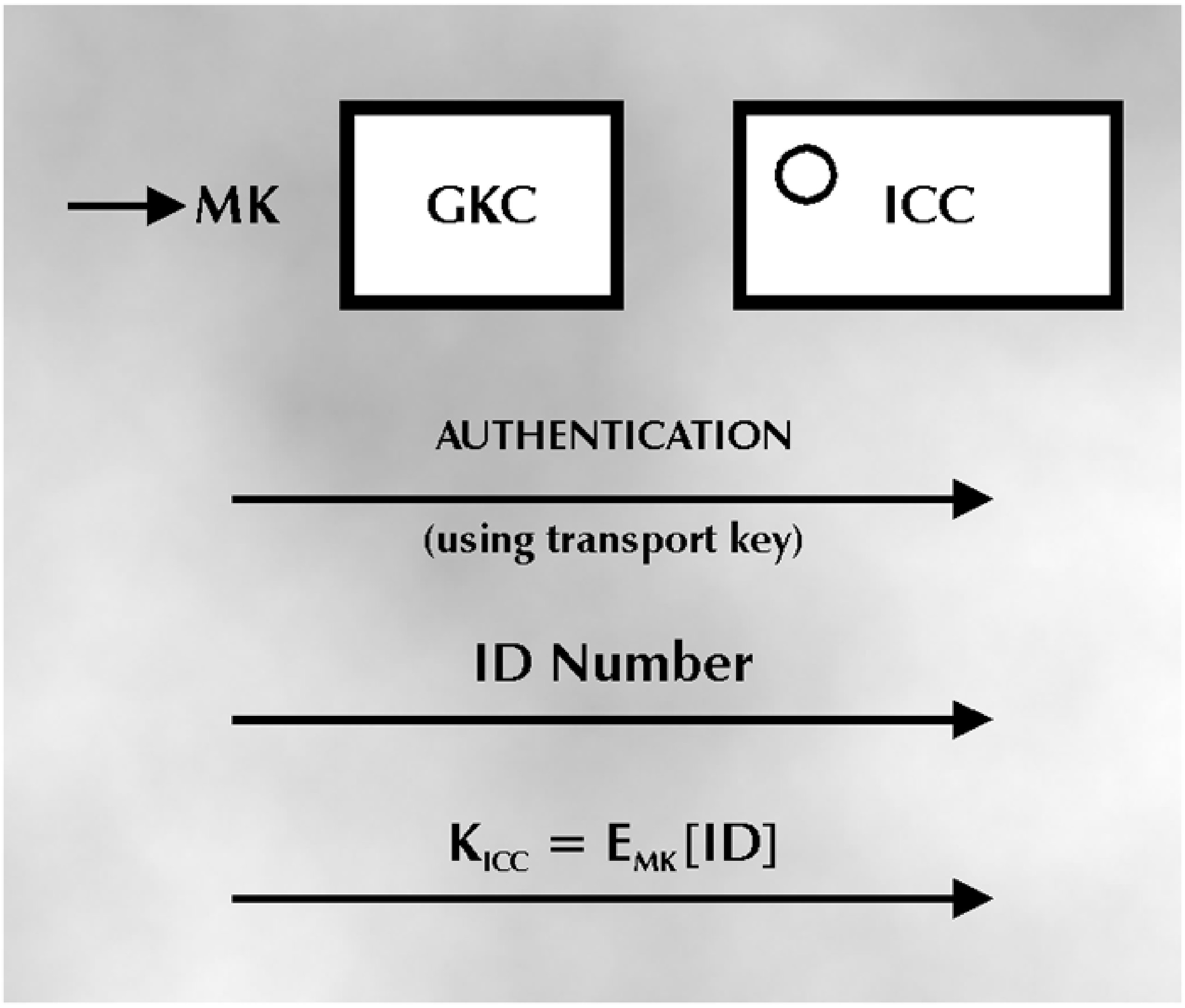


Figure 4: Symmetric Global Key Centre

We can now examine the primary key management operation for the security scheme. In fig.4 the GKC generates a unique key for each ICC which is derived from the reference number of the chip and the system master key. This reference number (shown as ID) and the unique key are loaded into the EEPROM of the chip. In general it is also necessary for bilateral authentication to take place between the ICC and the GKC which we will discuss in more detail in a later part of the tutorial.

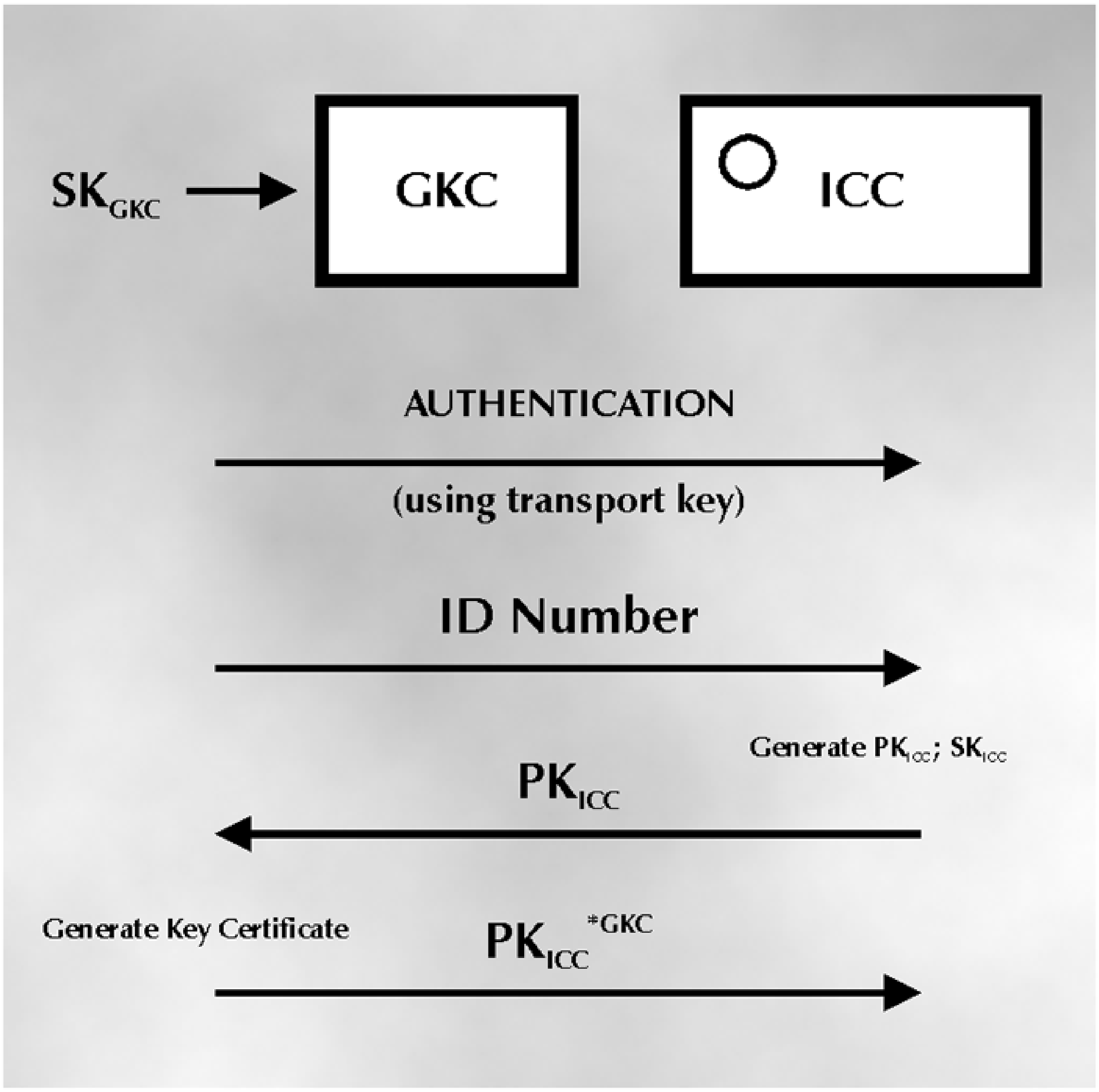


Figure 5: Asymmetric Global Key Centre

The asymmetric key management system is somewhat different and clearly there are many different ways of initiating such a scheme. In fig.5 we show an attractive approach where the ICC generates its own public key and matching secret key. The role of the GKC now relates to producing a key certificate to be stored in the ICC and presented to a corresponding entity as required. In this situation it is not necessary to reveal the secret key of the ICC to any party. The key is generated, stored and destroyed all within the particular ICC. There is however a price to pay for this security advantage. Apart from the relevant software code necessary to generate the public key/secret key pair there is an enormous performance overload at the GKC in the time required to generate the keys. Even with microcontrollers incorporating a numeric co-processors we need to allow about 30 seconds or so just for this initialisation operation. On a production line this parameter also limits the throughput to 120 cards per hour on a serial feed. Clearly this is a significant problem and one that needs a conceptual rethink of the ICC personalisation and initialisation process compared with the personalisation of existing magnetic stripe cards.

David Everett

Smart Card Tutorial - Part 17

First Published in January 1994

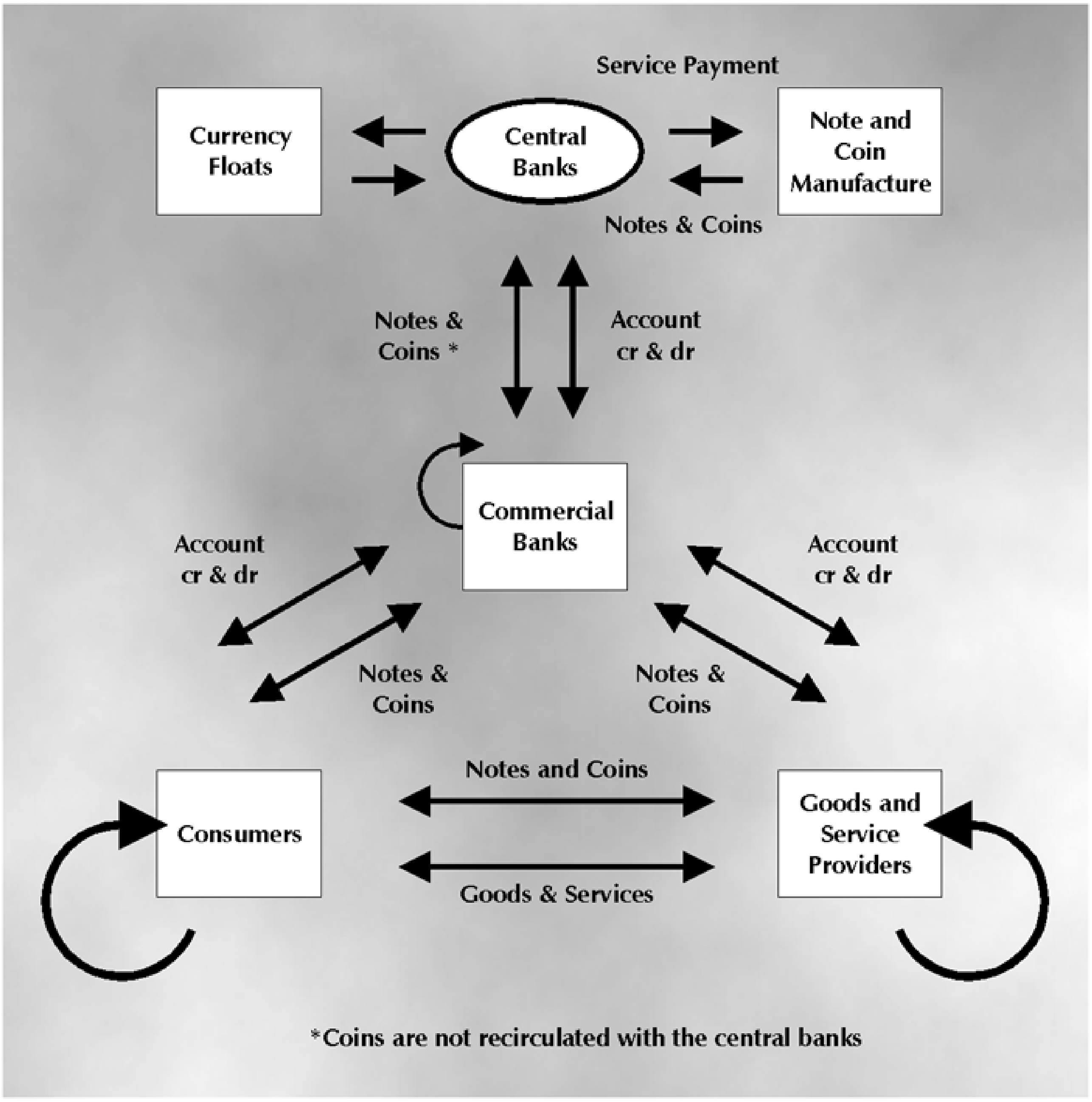
The Electronic Purse

The term electronic purse is used to describe a wide range of business scenarios where a Smart Card is used to represent the holding of a financial asset. In this part of the tutorial we will examine the more common situations and discuss the various models that arise. We can generalize these architectures in three ways,

a) Closed prepayment systems
b) Open prepayment systems
c) Electronic money

The business model is very different between these architectures and accordingly that leads to different requirements for the technical infrastructure that allows the business to operate. Before we start it is useful to hold in our minds the classical model for the use of cash as shown below. It is important to notice immediately that the model is multi currency. In the real world we are faced with a range of currencies that exist within the various entities defined. Any participant may be holding one or more currencies at any particular point in time. Equally obvious is the fact that these notes and coins are constantly recirculating amongst the various entities. The only exception here is the reluctance of central banks to accept coins which leads to a significant operational problem for the commercial banks. The distribution of notes and in particular coins is an expensive overhead.

The manufacture of currency is organised by the central banks. They are responsible for the security which underlies the use of their own particular notes and coins. In practice the actual manufacture of the notes and coins is contracted out to a manufacturer with the appropriate skills and security resources. The coins and notes are distributed on request to the commercial banks with the appropriate adjustment of accounts. At this point in time the notes and coins take on a value equivalent to their denomination which results in the creation of an asset float which relates to the total value in circulation. The central banks have a liability for the value of the float.



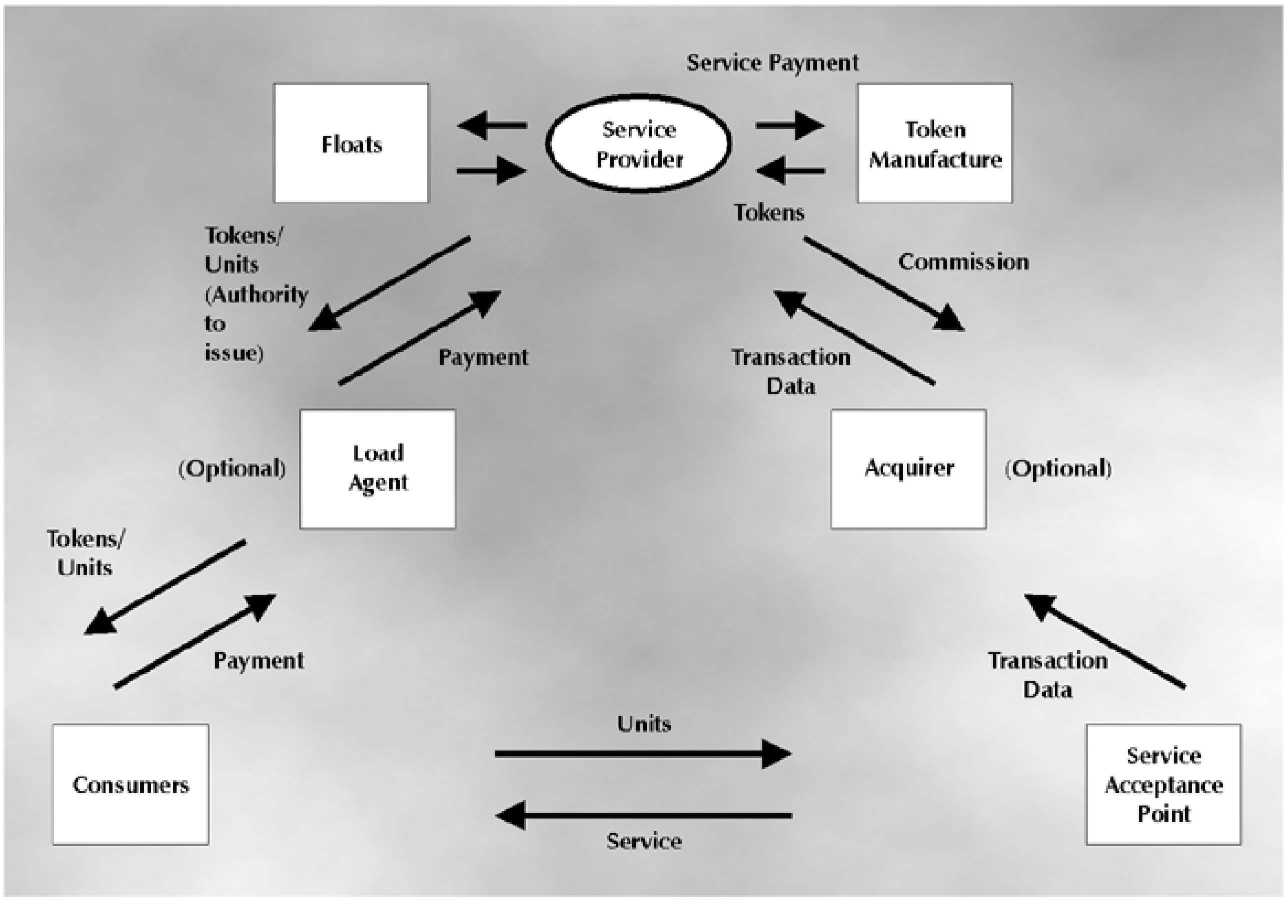
It is interesting at this point to reflect upon the matter of counterfeiting. The rules of the game are quite clear in that no entity wishes to accept a counterfeit note or coin. The central banks quite clearly will try to avoid the reimbursement of any counterfeit note in the distribution chain. If a consumer inadvertently accepts such a counterfeit then like a hot potato he will probably try to pass it on (I hope that I have not totally misjudged the morality of the average consumer). The commercial banks have the same problem but the morality issue is overpowered by the legal requirements. If the bank teller spots the note as a counterfeit then he will confiscate the note to the detriment of the customer. Whilst all this may seem academic it serves to highlight the security issues and also gives a first indication of the problems that surround the near perfect counterfeit note. As long as the central bank can detect the counterfeit then someone else in the chain will lose. Whilst few provable figures exist it is generally accepted that the American notes are one of the easiest major currencies to counterfeit. Such discussions often lead to an examination of the advantages that electronic money might offer in this respect.

Closed Prepayment Scheme

This concept is well established and used in most developed countries within the utility sector. The use of telephone prepayment cards is well known and other utility organisations such as gas, electricity and water are involved with the use of prepayment tokens. The model for this scenario as shown below is the simplest and technically the easiest to achieve. The first important point to notice is related to the flow of the electronic units. The concept of bidirectional flows has gone. The units flow from the service provider to the consumer and thence to the service point where they are collected. From the service point the necessary records flow back to the service point for reconciliation against the float. In general these units are not redeemable and the float always (assuming adequate security) shows a surplus. This is due not only to the time difference between acquiring units and using them but also because of losses by the consumer through inefficient use of the tokens. Analysis of throw away telephone cards provides an indication that this gain to the service provider can be substantial.

The next consideration here relates to the design of the token,

- passive
- active
- disposable
- reloadable

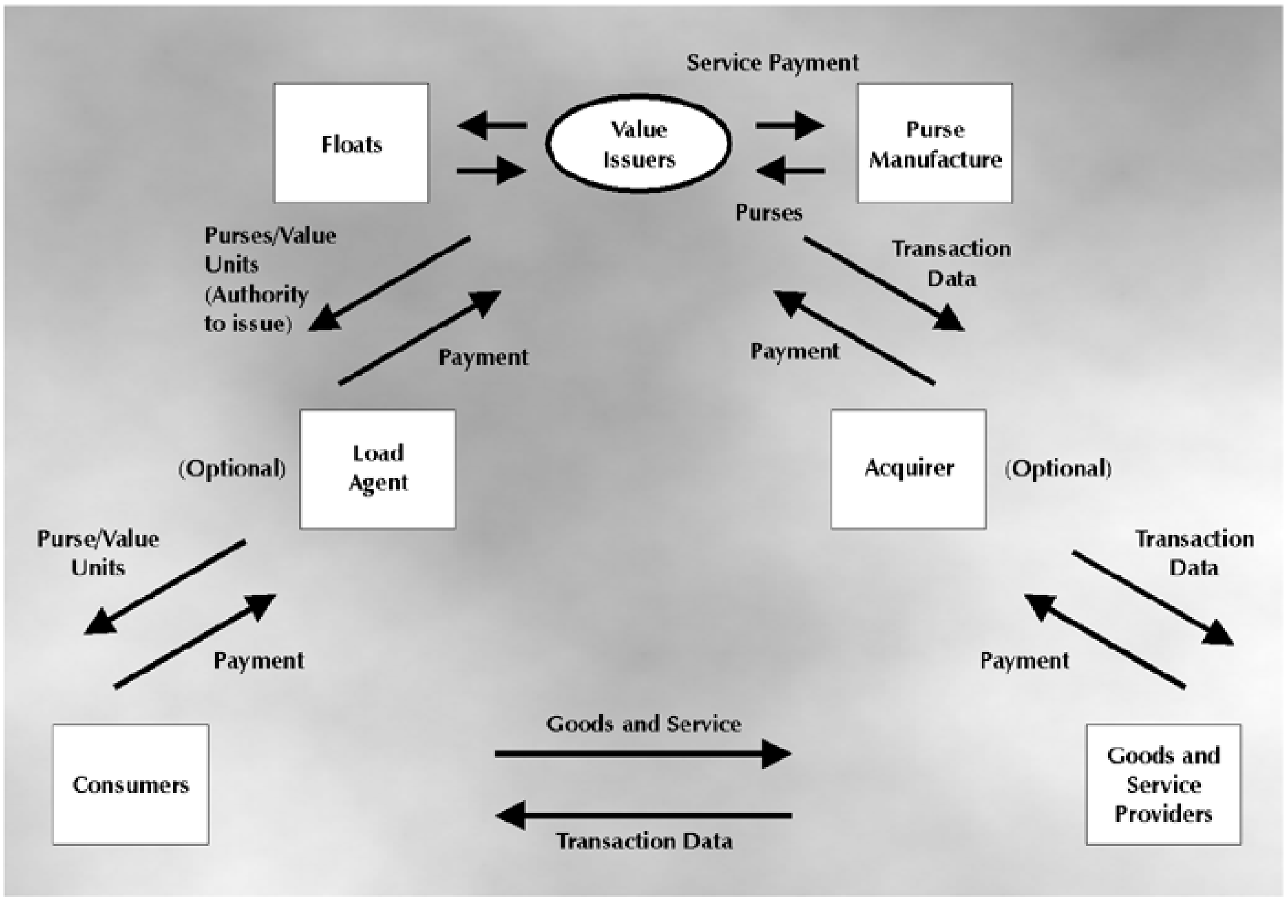


Most telephone cards are passive in that they store a data field reflecting the current value of the card. When the card is used this value decreases according to use. The security of a basic magnetic stripe card is really non existent since a perpetrator may easily copy the data stored on a new card and produce an arbitrary number of counterfeits. In Japan this is currently causing a major fraud problem which may result in the move to other techniques such as the Smart Card.

A basic integrated circuit (ICC) Card can also be implemented as a passive memory device and when used in this mode offers little advantage to the magnetic stripe card. A rewritable memory device using say EEPROM would be easier to defraud than a magnetic stripe card. Many of the memory ICCs however offer a protected memory mode where it is necessary to enter a PIN to unlock the memory access.

The problem here is that the access control mechanism is relatively primitive and by default must be common to all service points. It is difficult to imagine that these basic passive memory devices will ever offer adequate security except in very controlled environments. (Such as that offered by closed prepayment schemes). It should be noted here that telephones (for example) are permanently connected to the service provider which allows a range of security mechanisms not available in the more general off line environment.

The active token opens up more powerful security techniques and in particular the ability to achieve bilateral authentication. Clearly there is a significant cost increase in the use of a microcontroller ICC. As a rule of thumb the microcontroller card is likely to cost 2 - 3 times the cost of a memory card. In real terms for a mature volume scenario we are probably comparing 50p with about: £1-50.

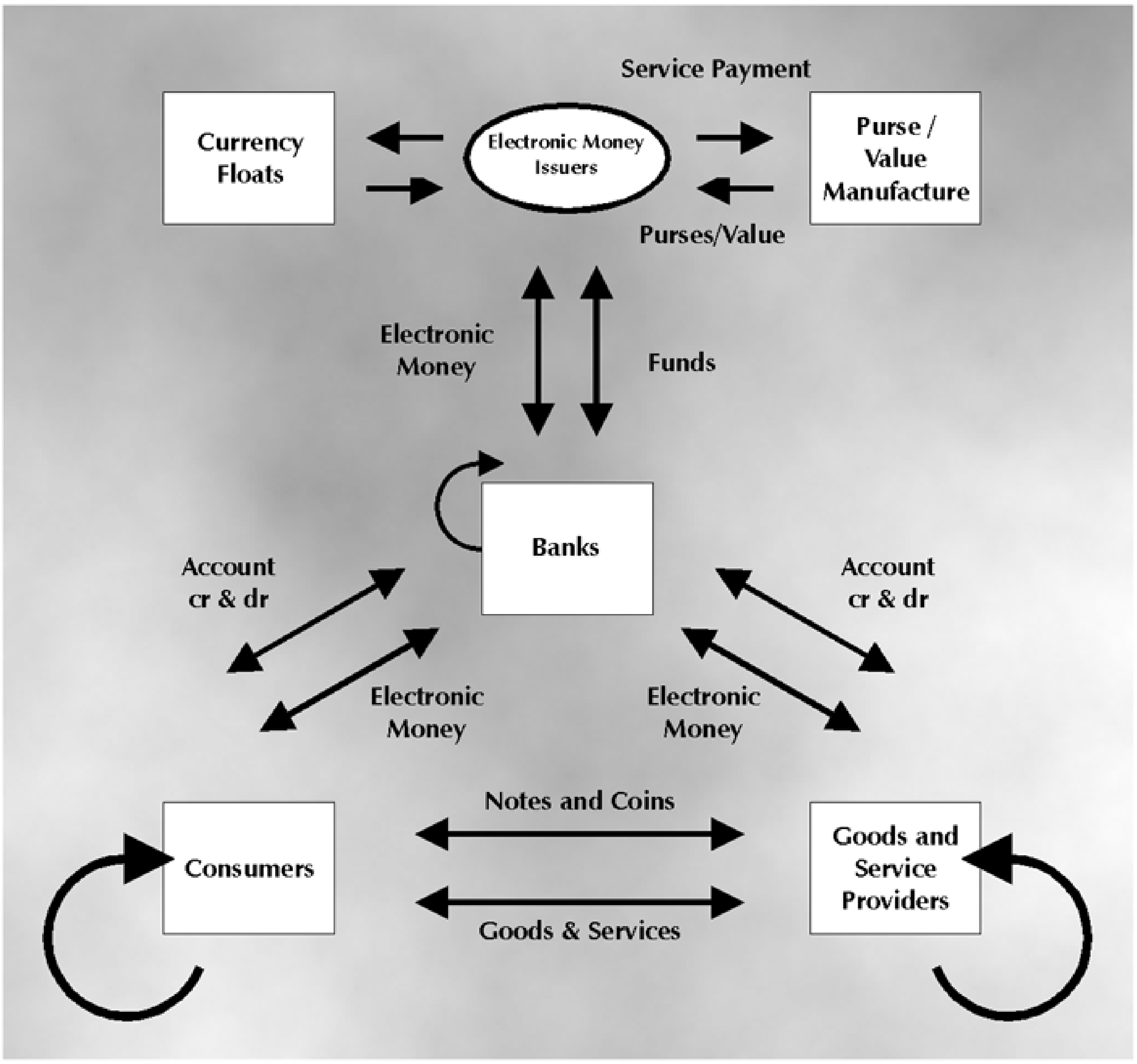


Whilst many utility card schemes use disposable cards there is no inherent technical reason why a reloadable card scheme should not be implemented. In many cases the business driver is for a disposable card and this is particulary relevant to telephone payment schemes. However these are significant changes taking place in the world of telecommunications which will probably change the picture. As mentioned previously the telephone service providers have the particular advantage of an on line link. This is not the case with other utilities such as gas and electricity. It should also be noted that it is still necessary to recover transaction information from the meter and it is this factor in particular that defines the overall technical architecture and its security requirements. What must be very clear is that the security requirements for a reloadable card scheme are more severe and therefore are intrinsically more expensive to implement.

Open prepayment systems

Flexibility is the antithesis of security. From a business point of view one would like to achieve maximum flexibility for the operation of a prepayment scheme and this is the core of our discussion when comparing an open prepayment scheme with an electronic money scheme.

A model for the open prepayment scheme is shown on page 15. There are in fact many similarities with the closed prepayment model. In this case the transaction records are acquired and transmitted back through the system to enable a clearing operation to take place. This is often compared as the electronic analysis of the traveller's cheque. In the model electronic value is shown moving in an anticlockwise direction. It is clear that this in no way maps our model of real cash. The security requirements are none the less stringent.



The model shows a general architecture for which there are a large number of possibilities in terms of commercial implications. Open prepayment schemes are still in their infancy and none have yet been applied on a international scale. The general implementation of such a model is predicated on a clearing between the value issue and as such the issuers of electronic value and its recovery requires the preservation of issuer identification. The optional load agents role arises due to the need to simplify the practical operation of the system. Neither the local agent or acquirer are intrinsic to such implementations but the mechanisms by which value clearing operates is likely to result in these roles which could of course be undertaken by a common agency.

Electronic money

On page 16 we show a model for electronic money. A comparison with the previous model for cash shows the similarity between the two approaches. With reference to the standard open prepayment model we can see instantly that the concept of clearing is removed if we assume that the electronic money for each currency is issued by one logical source. This is of course the case with real notes and coins.

The model also shows the same bidirectional flows for electronic money as for cash. In a general sense the electronic money circulates inter and intra the entities.

We announced in last month's SCN the Mondex electronic money scheme. It is clear that this is based on the primary model shown here.

The security architecture is quite different for each of these models. Next month we will develop a security model for these various purse models and develop an approach for the various implementation paths. It is however important to note that these models represent a general case. A particular implementation may invoke various restrictions to meet a particular business need. As an extreme case the removal of bidirectional flows on the electronic money prepayment model could lead to a commercial equivalent of the general open prepayment model.

Next month - Security and the electronic purse.

David Everett

Smart Card Tutorial - Part 18

First Published in February 1994

Security and the Electronic Purse

There are now a large number of national electronic purse initiatives and it's probably true to say that no two schemes are the same. The basic principles are however common and at this stage we will attempt to determine the core security requirements and see how they may be achieved. For our purpose we will take the simplest model shown in fig.1.

The model just shows the three principle participants, the purse provider, the purse holder and the service provider. These are the primary commercial entities involved in an electronic purse system and in each case a security module is used to effect the necessary transactions. By convention we will refer to the security modules of the purse provider and service provider as SAMs (Secure Application Modules). The purse holder's purse by definition forms the other security module.

There are three relationships by which electronic value may be manipulated in the system,

- Purse provider - purse holder
- Purse holder - service provider
- Service provider - purse provider

Accordingly the core function of the electronic purse scheme is to provide transactions that allow electronic value to be transferred amongst the participants. These transactions may be different for each relationship or could in principle be the same. It is the commercial relationship that defines what may or may not be allowed.

The primary security requirement is obvious but none the less leads to a fundamental determination of the security architecture,

Requirement (1) - value conservation

Value shall not be created or destroyed except in an authorised fashion.

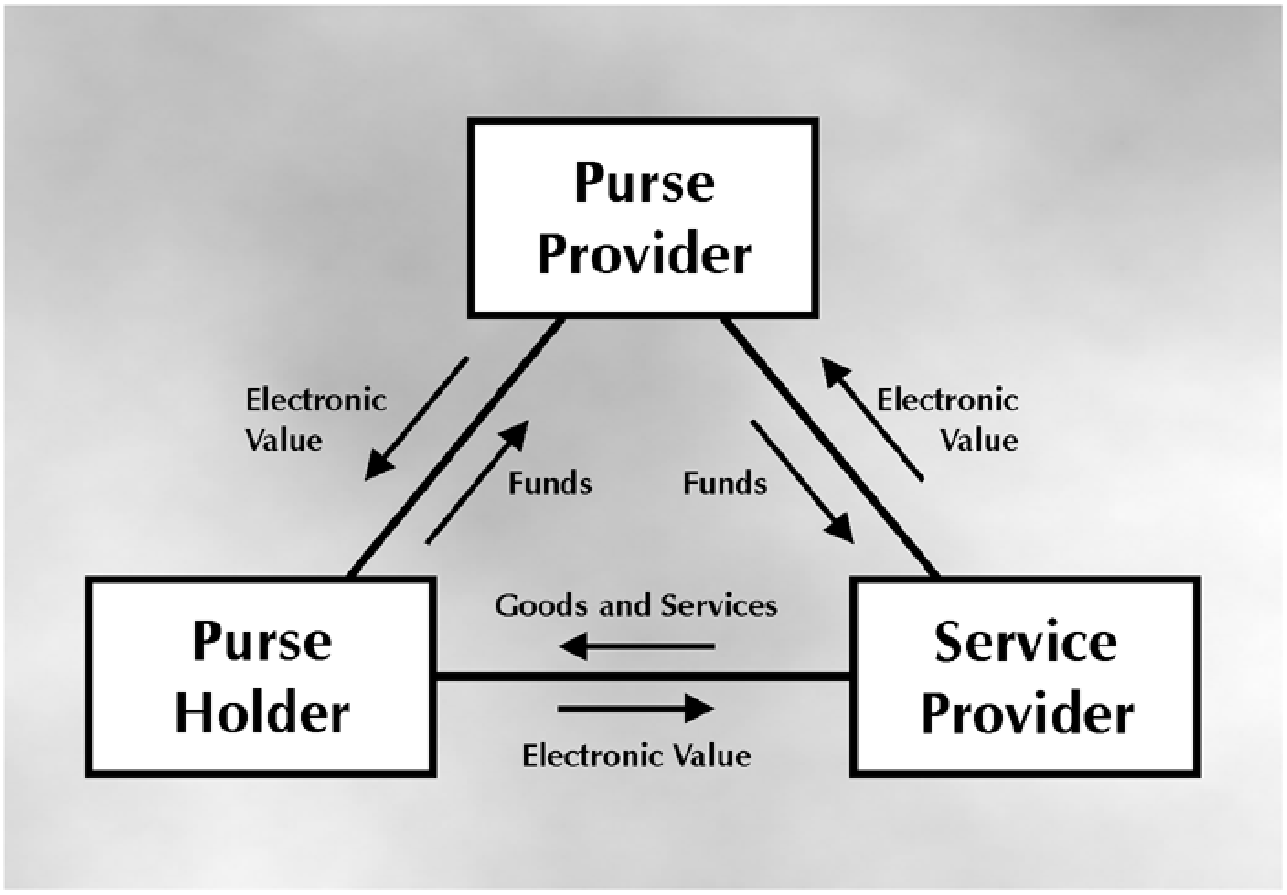


Figure 1: The Basic Purse Model

Clearly in this basic model only the purse provider is authorised to create and destroy value. If either of the other participants can create value then we are faced with a fraud scenario. If value is destroyed then one or more of the participants will suffer an economic loss. We can rewrite the basic commercial model by using its technical components as shown in fig. 2. This enables us to write the requirements necessary for value conservation.

a) The data integrity of the value stores must be preserved

b) The value transfer protocol must maintain value equilibrium (i.e total value before a transaction must equal total value after the transaction).

In a real environment neither of these conditions can be guaranteed. Components in the system may fail leading to a breach in either requirement. Clearly such failure must be minimised and in so far as is possible recovery mechanisms must be capable of resolving such problems.

Integrity of the value stores

However the purses and SAMs are designed it is clear that the value store will be implemented by some form of non volatile memory. In practice this will almost certainly be EEPROM memory. There are two principle technologies used for EEPROM, Flotox and MNOS. Both of these technologies can fail albeit by different mechanisms. All EEPROM technologies have an endurance capacity which is quoted by the manufacturers and is typically in the region 104 - 105 write cycles. Although this may seem adequate a busy service provider could invoke a thousand transactions per day. Assuming that the relevant data memory is only written once per value transfer we are still faced with a SAM life of 10 days if we take the lower of the range quoted above. Regardless of the design invoked for memory usage we must still allow for failure and it is clear that some form of data integrity function is necessary. Some implementors routinely use a four bit

ECC (Error Correcting Code) for each byte of EEPROM memory. There are other possibilities such as the maintenance of mirror data images. So far we have considered failure conditions but quite clearly it is necessary to be assured that it is not possible to alter the content of the value store by unauthorized means. It would be quite horrendous if users could alter the content of this store by subjecting them to some form of external radiation. Similarly one must be assured that the integrity of data is preserved within the actual mechanism of the EEPROM write cycle. In an earlier part of the tutorial we explained in more detail the operation of such memories.

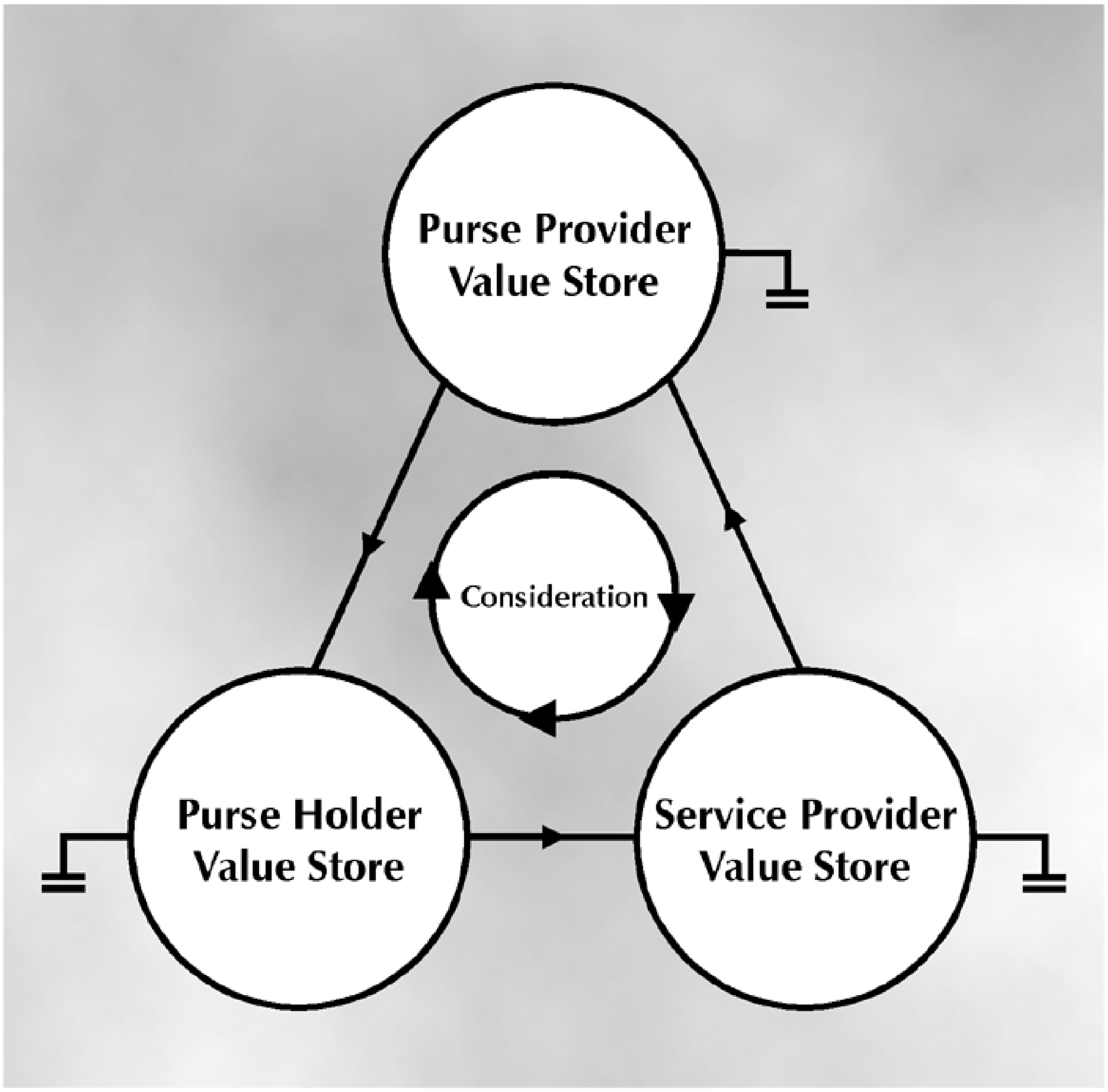


Figure 2: Electronic Value Component Mode

Value Transfer Protocol

The process of transferring value between the secure stores has been the subject of extensive academic research. There are a large number of options but most are based on the same security principles. The differences are in the cryptographic mechanisms used to achieve the necessary security services. The first requirement operates at a high level but is necessary to prevent potential fraud scenarios,

- The payer store must be decremented in value before the payee store is incremented in value.

Whilst the successful achievement of this requirement is necessary for value equilibrium (the converse is equivalent to unauthorized value creation) it leads to a number of value loss conditions in the event of some failure in the execution of the value transfer protocol. Accordingly the protocol needs sufficient auditability to enable recovery mechanisms to take place in the event of some failure.

The next three requirements are found in any electronic payment system,

- Entity authentication must be adequately assured
- Data integrity must be adequately assured
- Message replays must be prevented

We have defined that the value transfer protocol operates between two secure stores of value. Then in our simple model the fundamental payment mechanisms is made between the user's purse and the service provider's SAM. Thus the first requirement leads to mechanisms whereby the purse can authenticate the SAM and for the SAM to authenticate the purse. It is clearly mandatory for the SAM to authenticate the purse. The authentication of the SAM by the purse is necessary as part of the proof of payment security service and can prevent mischievous loss of value. A sequence of messages may be established which implements these security services as shown in fig. 3. Three separate phases are shown,

- Entity authentication
- Value payment
- Proof of payment

It is assumed here that the terminal plays no part in the cryptographic security services. Thus the entity authentication mechanism takes place between the purse and the SAM. The value payment transaction also takes place between the purse and the SAM whilst the transaction record proof may optionally be sent from the SAM to the purse. The requirement for this is dependant on the commercial structure of the electronic value scheme. If the individual transactions are not recorded by the terminal then the proof of payment can be completed by passing this message back to the purse. In all cases it is necessary to achieve two additional requirements,

- The purse holder should authorise the transaction.
- The purse holder should be adequately assured of the correctness of the transaction event.

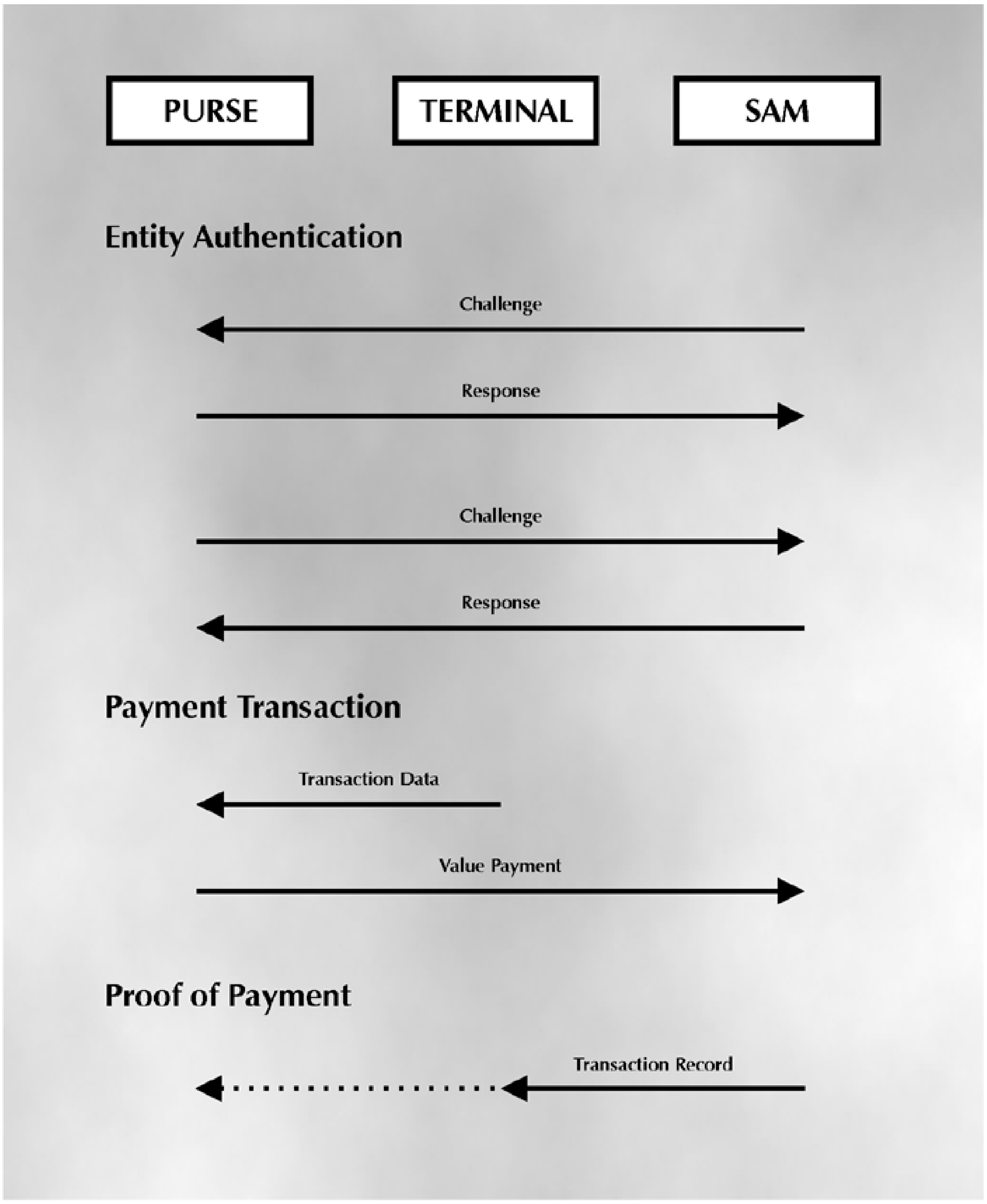


Figure 3: Standard Payment Transaction

In practice these additional requirements place a security burden on the terminal to adequately allow the purse holder to control the payment transactions. At the very least he should be assured that what he authorises is actually what takes place.

By combining the security mechanisms we can reduce the message set into a generalised protocol as shown in fig.4. Here we have shown three core messages for the value transfer protocol,

- An authentication/transaction challenge
- A value payment / authentication response
- A transaction record

The first two messages are always between the purse and the SAM. The transaction record may be logged by the terminal for reconciliation with the purse provider or may be truncated by the SAM. Optionally the transaction record may be sent to the purse as an acknowledgement message.

Clearly these three messages must be cryptographically protected by some form of digital signature. This may be a true signature using an asymmetric cryptographic algorithm or some form of cryptographic check value using a symmetric algorithm. The important difference here is that in the case of the symmetric algorithm the terminal is unable to verify the cryptographic mechanism.

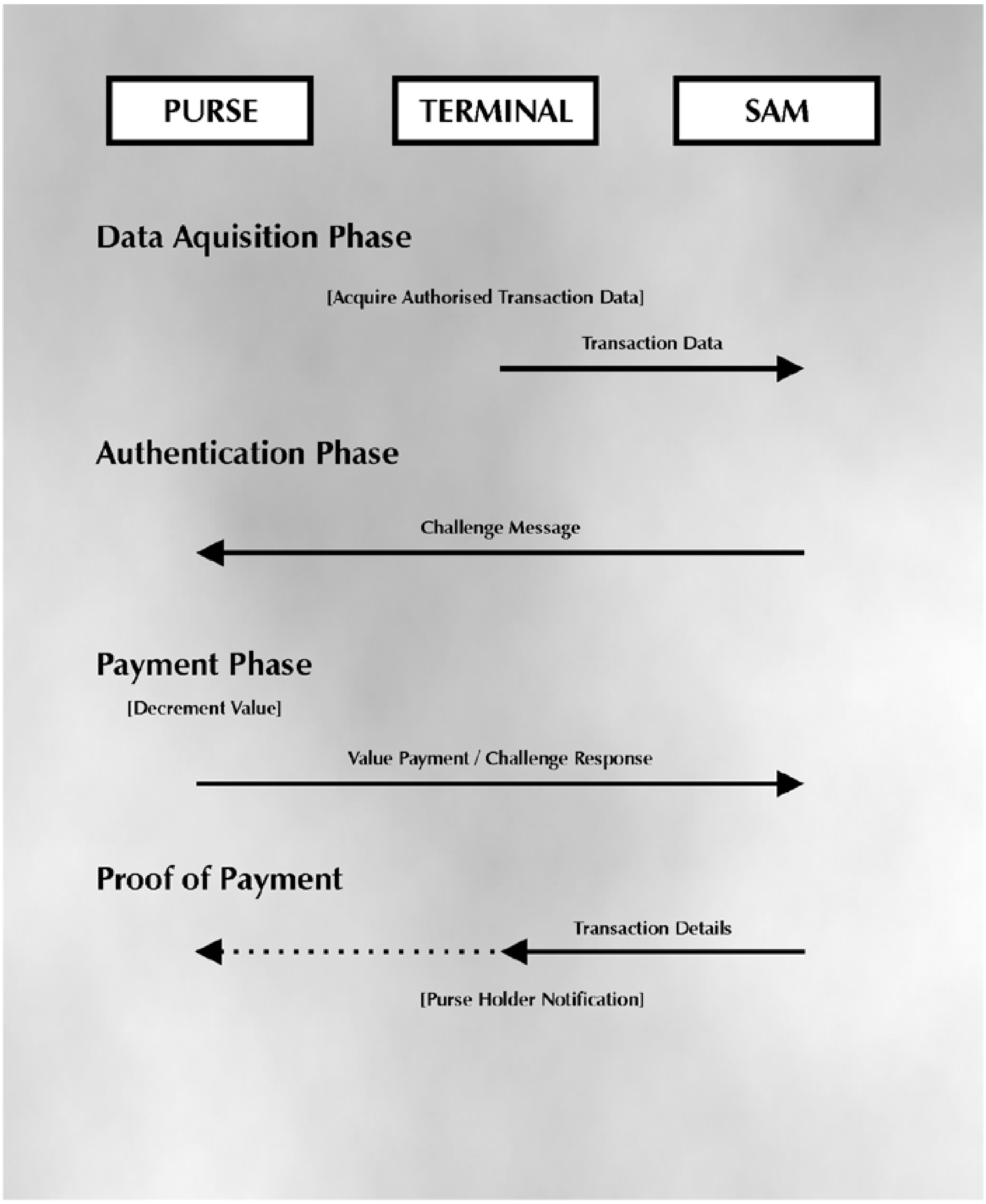


Figure 4: Combined Authentication / Payment Messages

For the value transfer protocol to offer adequate security the following requirements are necessary,

- The challenge message and the value response must be unique to the purse and SAM.

Apart from the transaction data the challenge message should incorporate the unique identities of the SAM and purse and should incorporate a transaction unique sequence number. The value response should incorporate the same data. By such means duplicate payments can be avoided. The transaction data may include information relating to time and date but it would be generally unwise to use such data as part of the security mechanism because of the difficulty of controlling the terminal data. The terminal operator needs to adjust such data for the normal operation of the terminal and therefore cannot be prevented from abusing such data.

David Everett

Next month: Anonymous Payment Schemes.

Smart Card Tutorial - Part 19

First Published in March 1994

Anonymous payment schemes

Those people involved in the marketing of electronic cash systems will need little persuasion on the product benefits of conventional cash. The simplicity and flexibility of payments in shops or even between individuals is hard to better. One of the obvious features of cash is its anonymity. In general you can make payments for goods or services with no audit trail that would lead a third party to your identity. Whilst consumer needs for such anonymity must vary we can all imagine situations where such characteristics are desirable if not essential. The classical electronic purse scheme as we have discussed previously leads to a rather effective audit trail on consumer spending. Lets look at this model again as shown in fig. 1.

The conventional electronic purse involves three processes,

- value load
- payment transaction
- transaction settlement

Each of these steps involves a number of audit trails. The issuer normally provides electronic value to the consumer against an account debit. Clearly the issuer must identify the account holder and obtain his authorisation for the transaction. We may also assume that any electronic purse instrument will contain a unique identification number. We can easily arrange for the relationship between the purse and the account holder to be independent but in practice they are highly likely to be linked. Accordingly the issuer can and almost certainly would build a transaction log that shows the loading of a particular purse against an individual account.

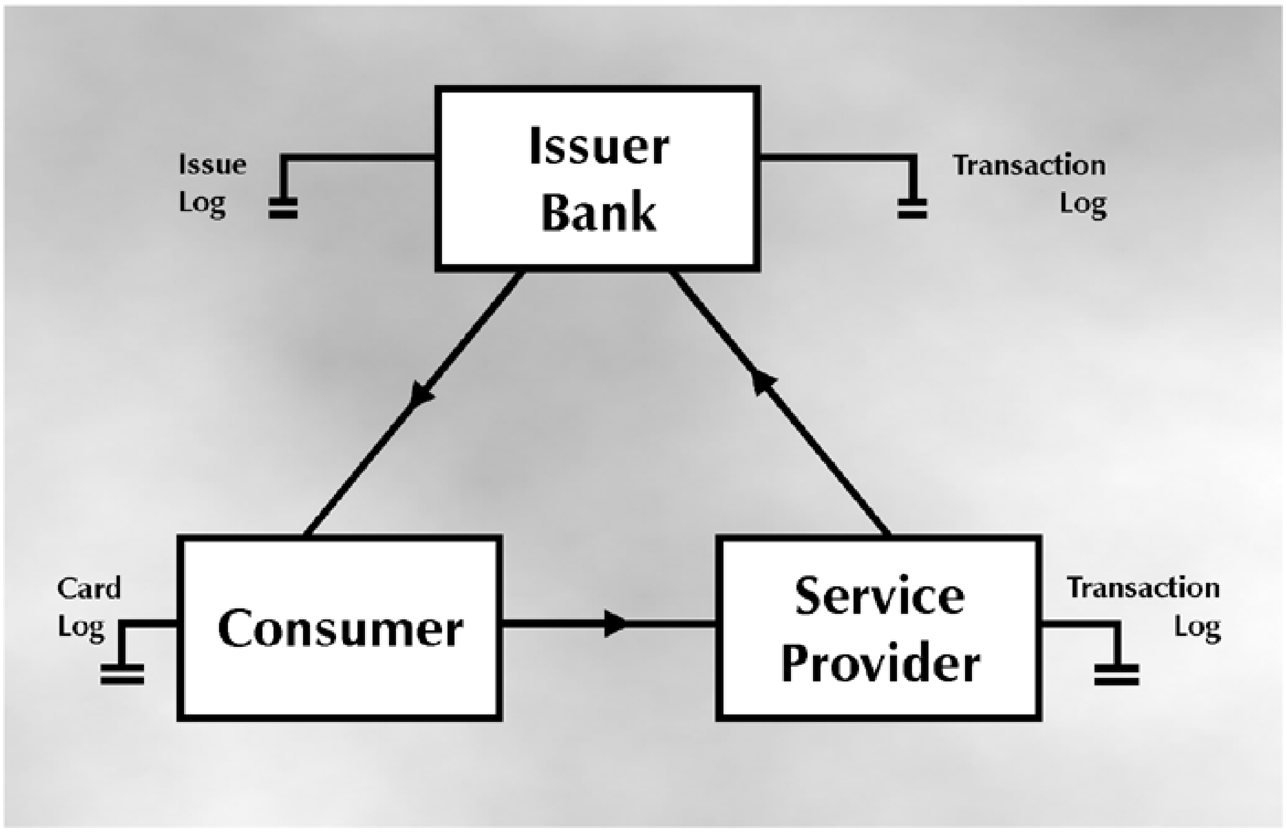


Figure 1: The Classical Audit Trail

When the consumer uses his electronic purse to make payments for goods and services the service provider will make a record of the transaction as the basis of subsequent payment by the issuer. This transaction will need to include a record of the purse identity in addition to the value of the transaction. This transaction log however is totally anonymous in that it should not contain any correspondence with the card holder's identity.

When the service provider submits the transaction record for payment by the issuer the data record will contain the unique identity of the purse which the issuer can relate not only to the account holder but also to the particular service provider and for the value of the payment.

It is quite clear that the information collected by the issuer is sufficient to build an information data base on the account holder's spending behaviour. This has caused concern amongst many system designers but the problem has been addressed in some detail by the mathematician David Chaum who has produced a novel solution to this problem.

The essence of David Chaum's technique can be best understood by a simple allbeit some what incomplete analogy. Let us consider a method for making a $1 payment for goods in a retail shop. The stages in the process are shown in fig.2.

The consumer takes a plain piece of paper that will become his certified $1 note. He prepares an envelope that contains this plain piece of paper along with a carbon copy paper. He then takes the sealed envelope to the bank who stamp the envelope with their certified $1 seal. The imprint of course ends up on the plain piece of paper contained inside the envelope. The bank will take payment for the use of the $1 seal but cannot see the plain piece of paper inside the envelope and therefore cannot subsequently identify the paper but can recognise the imprint of their seal.

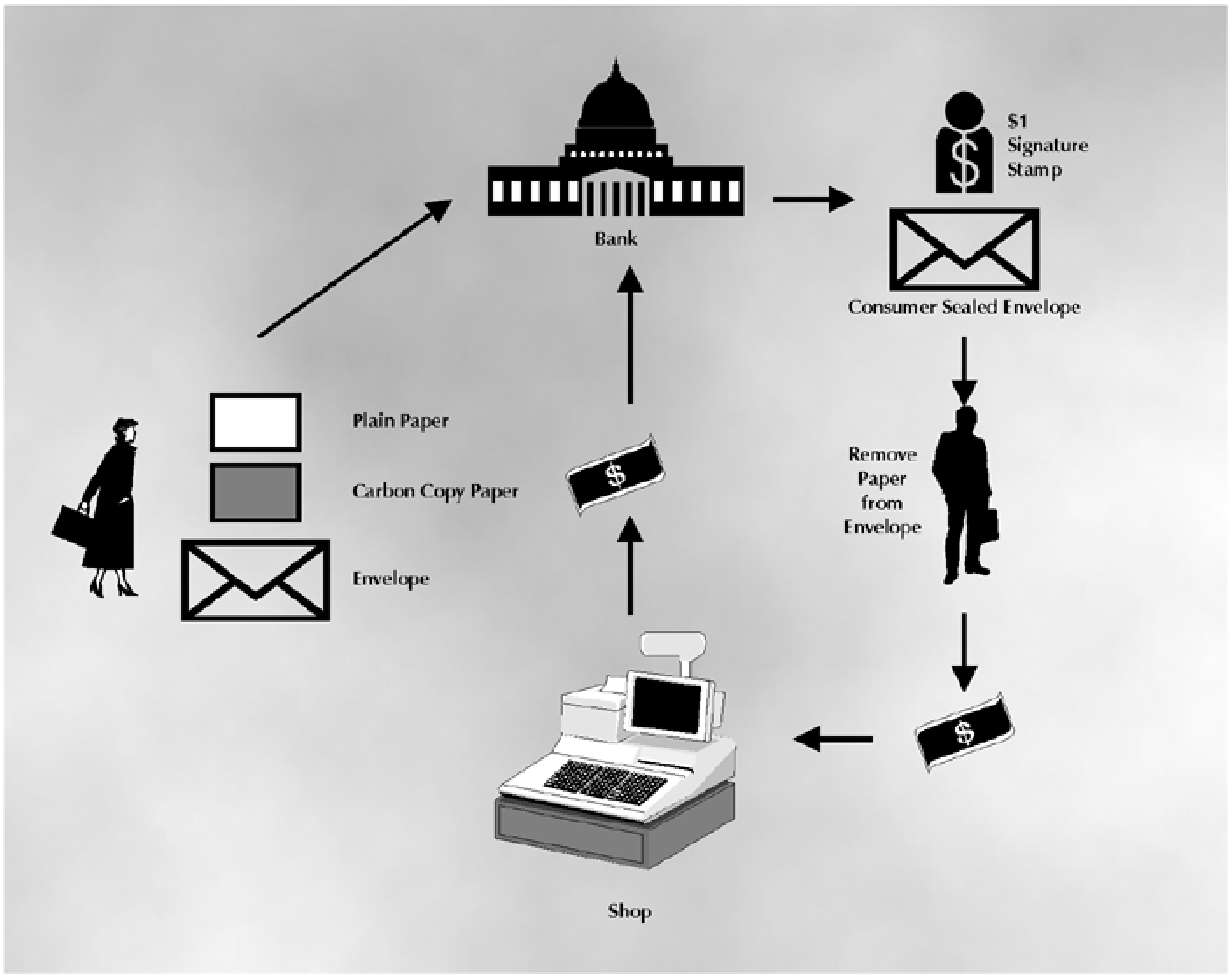


Figure 2: David Chaum's Anonymous $

Once outside the bank the consumer unwraps the envelope to remove the piece of paper that now has the bank's imprint for $1. The consumer takes the stamped paper to the shop which is able to recognise the bank's seal and offer goods in exchange. When this piece of paper is presented to the bank by the retailer then the bank can recognise it's seal and make payment in exchange to the retailer.

Chaum's mathematical equivalence can be explained by using an RSA digital signature. The main principle to remember is that the checking of a signature is made by comparing data with some mathematical transformation of that data. We can recall from previous discussions in the tutorial for an expression of the RSA algorithm using a public exponent of 3 as follows,

- $S = M^d \text{ Mod } N$ (i.e $\sqrt[3]{M} \text{ Mod } N$)
- $M = S^3 \text{ Mod } N$

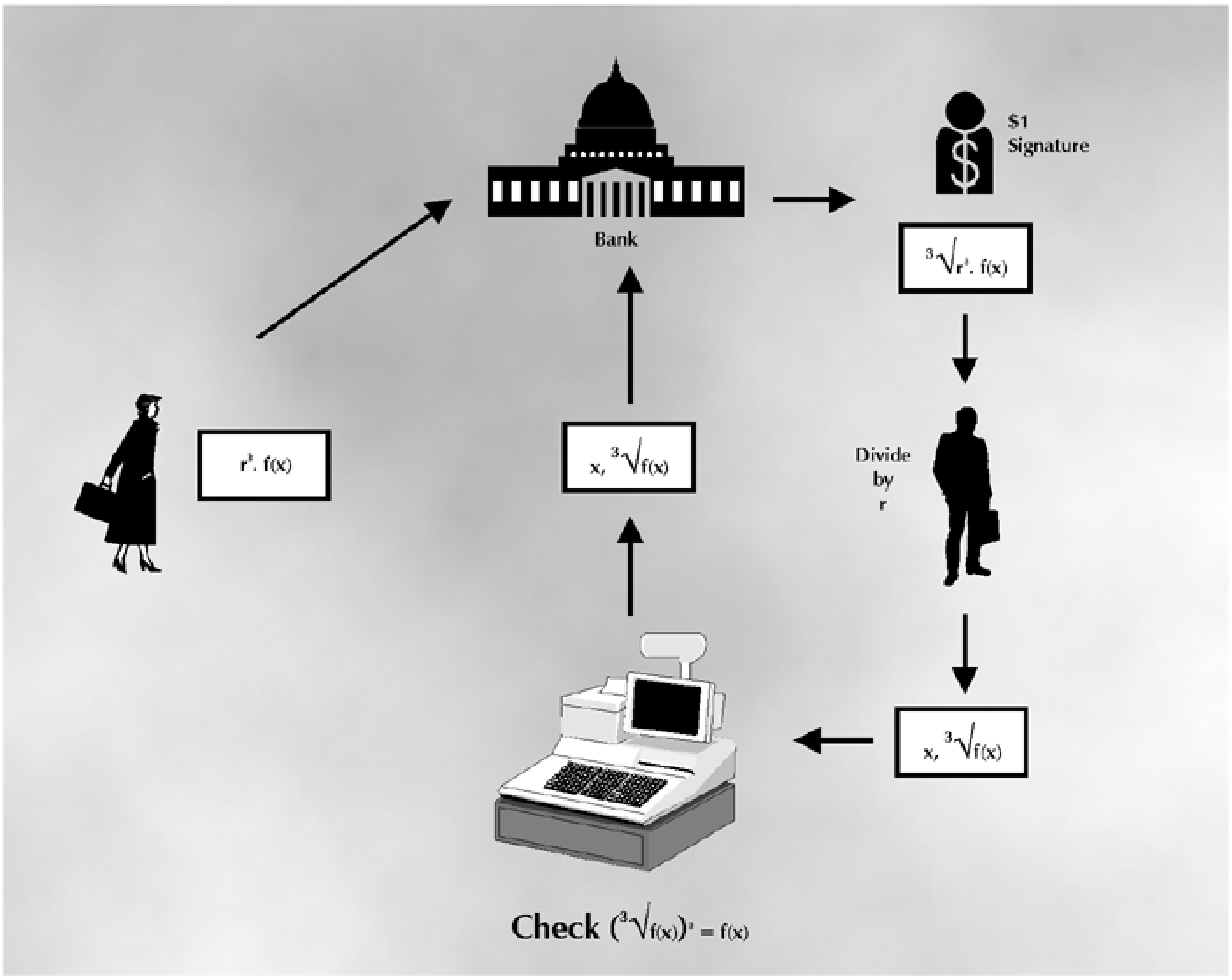


Figure 3: Chaum’s Electronic $1 Bill

The signature (S) here is effectively the cube root of the message (M) created using the secret key d. N is the common public modulus.

The mathematical process proceeds as shown in fig. 3. All operations are assumed to take place modulo N although this is not shown in the figure for clarity. Instead of a piece of paper the operations now take place using numbers and mathematical transformations. The consumer chooses two random numbers r and x and forms r^3 . f(x) where f(x) is a one way function of x. He takes this number to the bank which applies it's signature process thereby calculating the cube root. The number returned to the consumer is therefor equivalent to $r \sqrt[3]{f(x)}$. The point to notice here is that the bank knows neither x or r since only the product r^3 f(x) was supplied.

The consumer can now divide the number supplied by the bank by the originally chosen value of r to recover $\sqrt[3]{f(x)}$. When he goes to the shop he gives the retailer both x and $\sqrt[3]{f(x)}$ which are now equivalent to the $1. The retailer can check the authenticity of this number by cubing $\sqrt[3]{f(x)}$ to see if it corresponds to f(x) (i.e a conventional RSA signature check).

The retailer subsequently sends x and $\sqrt[3]{f(x)}$ to the bank which can also check for its authenticity. However there is no way that the bank can make any correspondence between this signature and any of it's account holders. Therefor the transaction is truly anonymous. You can see that it was the use of the random number r that disguised the number provided by the consumer. This number was subsequently removed after the banks signature process. In the analogy this compares to the use of the envelope.

There are of course a number of short comings in the simple process described here and it is the ingenuity of Chaum's work in solving these problems that is particularly interesting.

We have mentioned previously that the principle weakness of a digital signature is the ease with which a fraudster may take a copy and therefor issue a duplicate. It is readily apparent that the retailer could phone the bank and ask if the particular signature in his hand has already been presented. However in practice this is not tenable with the concepts of electronic payments by an electronic purse.

The solution proposed by Chaum solves the problem in a different way and results in a scheme whereby the identity of the consumer will be revealed if he attempts to spend the signature twice. This rather elegant technique is based on a definition of the number given by the consumer to the bank for signature. In a simplified form we will show the principle of the detection method as follows,

the consumer computes a vector (say 20 elements) for signing by the bank as,

- $[\ r_i^3 \ .\ f(x_i\ ,\ y_i)\ ]$
- Where $x_i = g(a_i)$ and $y_i = g(a_i \oplus \mathrm{ID})$ [$\oplus$ = exclusive OR]
- Note: if $z = a \oplus \mathrm{ID}$ then $z \oplus a = \mathrm{ID}$

the a_i are randomly chosen and g() is another one way function. ID is the consumer account reference number. The signature created by the bank is now based on the product of these vectors.

- $\Pi\ \sqrt[3]{r_i^3}\ .\ \sqrt[3]{f(x_i\ ,\ y_i)}$

as before the consumer can remove r which Chaum refers to as a blinding factor (i.e. the anonymity factor).

When the consumer comes to spend the signature in the shop the retailer issues a random challenge vector [ci] for each value of i. The response from the consumer is as follows,

- If $c_i = 0$ send a_i and y_i
- If $c_i = 1$ send x_i and $a_i \oplus \mathrm{ID}$

This enables the retailer to check the signature and the correspondence of the responses. All the information including the responses are sent to the bank by the retailer in order to claim his payment.

Now lets look at what happens if the consumer tries to spend the signature twice. The next retailer will issue another random challenge vector and statistically we may expect that at least one of the challenge vector elements c_i will be complimentary. This means that when the bank receives the duplicate signature it will also have the responses from both transactions and for at least one case will have both a_i and $a_i \oplus \mathrm{ID}$. This will enable the bank to recover ID the consumers account number. In the full method additional random numbers and checks are made to stop the consumer cheating the bank on it's ID when the signature is first created.

There is another little problem with the method proposed so far. We have always assumed that the payment signature is for a $1 bill. In practice of course we need total flexibility on payment value. Well, the solution here is analogous to that above for revealing the account identity. In this case the challenge vector includes a representation for the value of the transaction. There is however a practical complication in that the initial signature generated by the bank needs to be based on a maximum value for that signature. If the consumer account is debited for that full amount then a reconciliation process will need to take place when the signature (with its lower value) is presented by the retailer to the bank. The whole process of anonymity will be spoilt if this took place on an individual signature basis.

The solution proposed by Chaum is for the consumer to obtain sets of checks from the bank so that the refund mechanism operates against the whole set. By this means the bank will be unaware of the amounts for the individual transactions. In order for this to work the bank's signature process operates with two signature moduli which enables the necessary blinding to take place.

We have gone all the way through this part of the tutorial without mentioning Smart Cards. Although we have never said so we have of course assumed that the consumer undertakes all the operations described here using a Smart Card which is his payment token. An interesting point to note about Chaum's method relates to the security requirements for the Smart Card. The only secret key operation is that undertaken by the bank and as such the bank cannot be defrauded because it will only pay on each signature once. The main security risk therefore relates to the use of duplicate signatures which in it self is not totally dependant on the security of the Smart Card. Under these conditions the vulnerable party is the retailer who needs to be assured that he is receiving the authentic signature. In his papers Chaum has proposed that for high value payments the retailer could phone the bank for signature authorisation.

David Everett

Next month - Smart Cards and cryptographic key management.

Smart Card Tutorial - Part 20

First Published in April 1994

Smart Cards and Key Management.

We have discussed the use of Smart cards for implementing various security functions such as data integrity and authentication. There is however an even more fundamental role for the Smart Card in the implementation of key management architecture. All cryptographic schemes require the use of a key transport module by which means the secret security keys can be distributed in the system as required. We should note immediately that any security scheme relies on the use of trusted entities and the necessary procedures for their use. In this part of the tutorial we will examine the establishment of security in a network of Smart Cards. The key management scheme involves the four phases of the security keys, generation, storage, distribution and destruction. Any such scheme involves the use of a trusted key centre and the necessary key hierarchy. The principles are the same for both symmetric and asymmetric cryptography although the mechanism and the practicality of the operation can vary considerably. For the purpose of our discussion we will assume that the Smart Card acts as a tamper resistant module with adequate protection against physical and logical attack. Key schemes are always arranged in some hierarchy for which the root of the key structures is the master key. Such schemes may involve three or more layers but for our discussion we will consider only a two layer scheme as shown in fig. 1. The principle may easily be extended to additional layers.

Key Generation

The root of the key hierarchy is defined as the master key from which all other keys are devised. Clearly this is the critical key in the system and must be non deterministic. It goes without saying that we should not allow any one person to obtain knowledge of the secret key. However human beings are intimately involved in the key management process and we must therefore ensure that we can enforce the necessary procedures to maintain secrecy of the keys.

Where does this master key come from? Well to be non deterministic it clearly needs to be a random number or to be derived from a random number. This random number must of course remain secret for all time. The subject of random numbers occupies the minds of academics the world over, which the reader may correctly assume confirms the difficulty of the subject. There are two sorts of random number, pseudo and real. A real random number generally uses the properties of physical science (such as shot noise from a semiconductor device) that can produce an endless stream of uncorrelated numbers. The pseudo generator uses some mathematical algorithms to generate a stream of random numbers. John von Neumann first proposed such a method for computers using what was called the middle square method. Let's assume we wish to calculate a string of 5 digit random numbers then starting with 12345, we square to give 15[23990]25 where 23990 is squared to give the next number 57[55201]00. This operation is repeated as necessary. Unfortunately many such simple generators have unacceptable weaknesses. In particular they may lead to short sequences before they repeat. For the middle square method a particular problem arises when the middle digits becomes zero since there will perpetuate a string of zero's. The reader is referred to Knuth (Semi number algorithms vol.2) which is the classical treatise on this and many other mathematical topics relating to security for more information.

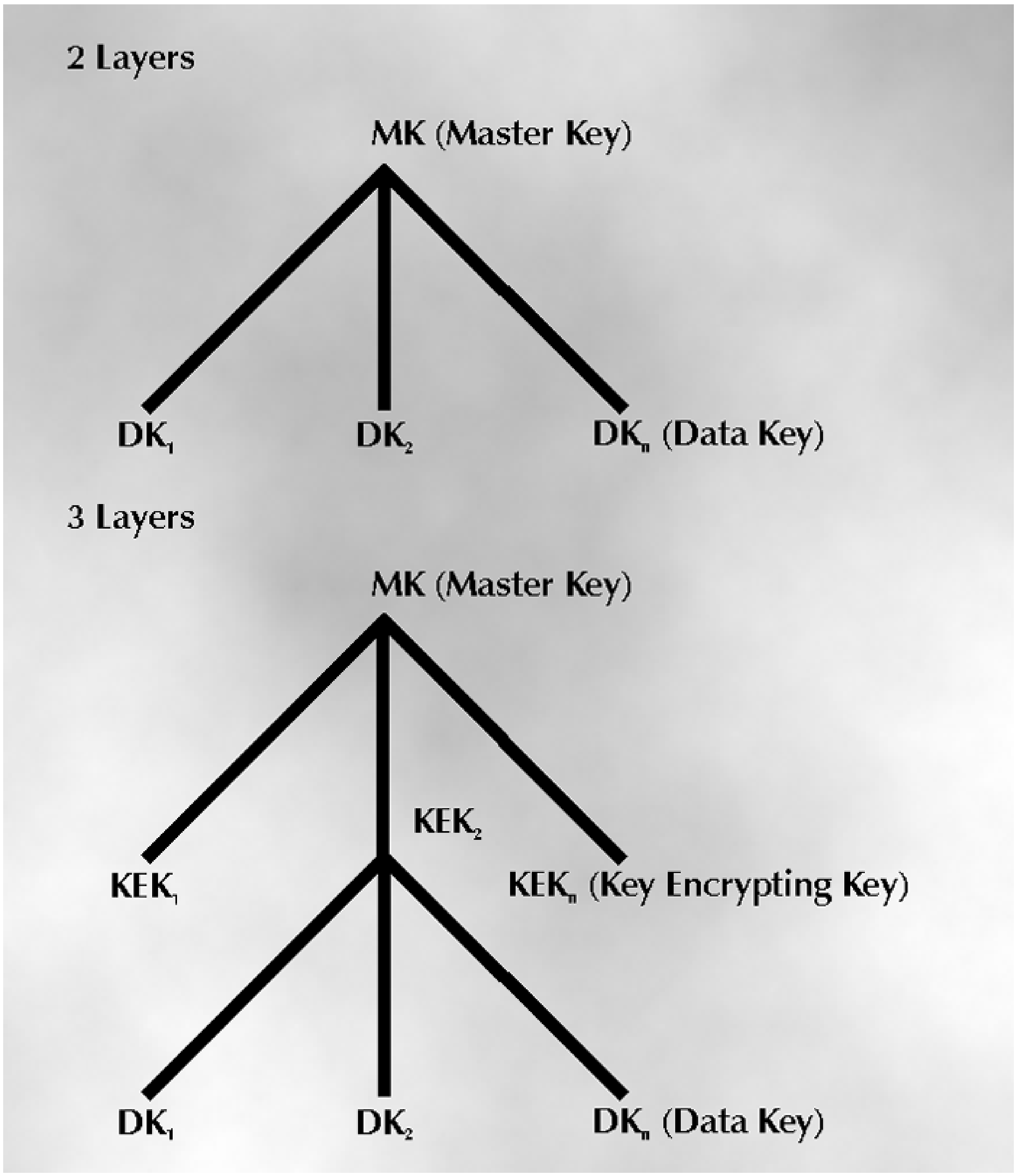


Figure 1: Key Hierarchies

Although von Neumann's method is not suitable as a pseudo random generator there are many other choices. In particular we could use the DES cryptographic algorithm which by definition must be non deterministic. If it were otherwise it would be totally unsuitable as a cryptographic algorithm. A way of using DES would be to take our first number as the key and then consecutively encipher the full range of input numbers (each of 64 binary bits). This will generate a non repeating sequence of 2^{64} numbers.

The observant reader will already have noticed the deliberate oversight, where did that first number (in our case 12345) come from? Clearly this should be equally non-deterministic or the whole sequence will be exposed. Yes, we also need a random number ‘Seed’ to start the sequence.

The problem is not actually circircuital since we only need in practice to find one good random number and a good pseudo random number generator. For all practical purposes our pseudo random number can do the rest. Several methods are widely used of which the favourite on a computer is to use the time differences generated by the user in entering a sequence of key strokes.

Let us now refer to our problem of starting off the root of the key management hierarchy. First of all we need to produce a tamper resistant module that contains our master key. For the moment we will assume a symmetric scheme such as an DES. In this case our master key (MK) is just the random number. Now either the master key has to be entirely calculated within the security module or we have to calculate it outside the module and inject it. It is clearly preferable to calculate the random number inside the module but in practice it is often inserted. The problem here is that just having one copy of the master key does not allow us to regenerate the key for fall back in the event that our security module fails. We obviously have to devise techniques to handle this problem. The oldest method for handling this problem is to generate the master key in parts. For example we can use three trusted officers each of whom have 1/3 of the key. Each officer enters his part of the key into the module which then calculates the master key as say, the exclusive or of the three parts. Each part is stored secretly by each of the officers. The security here is dependant on the integrity of the three officers, the storage of each of the key parts and the randomness of the key parts.

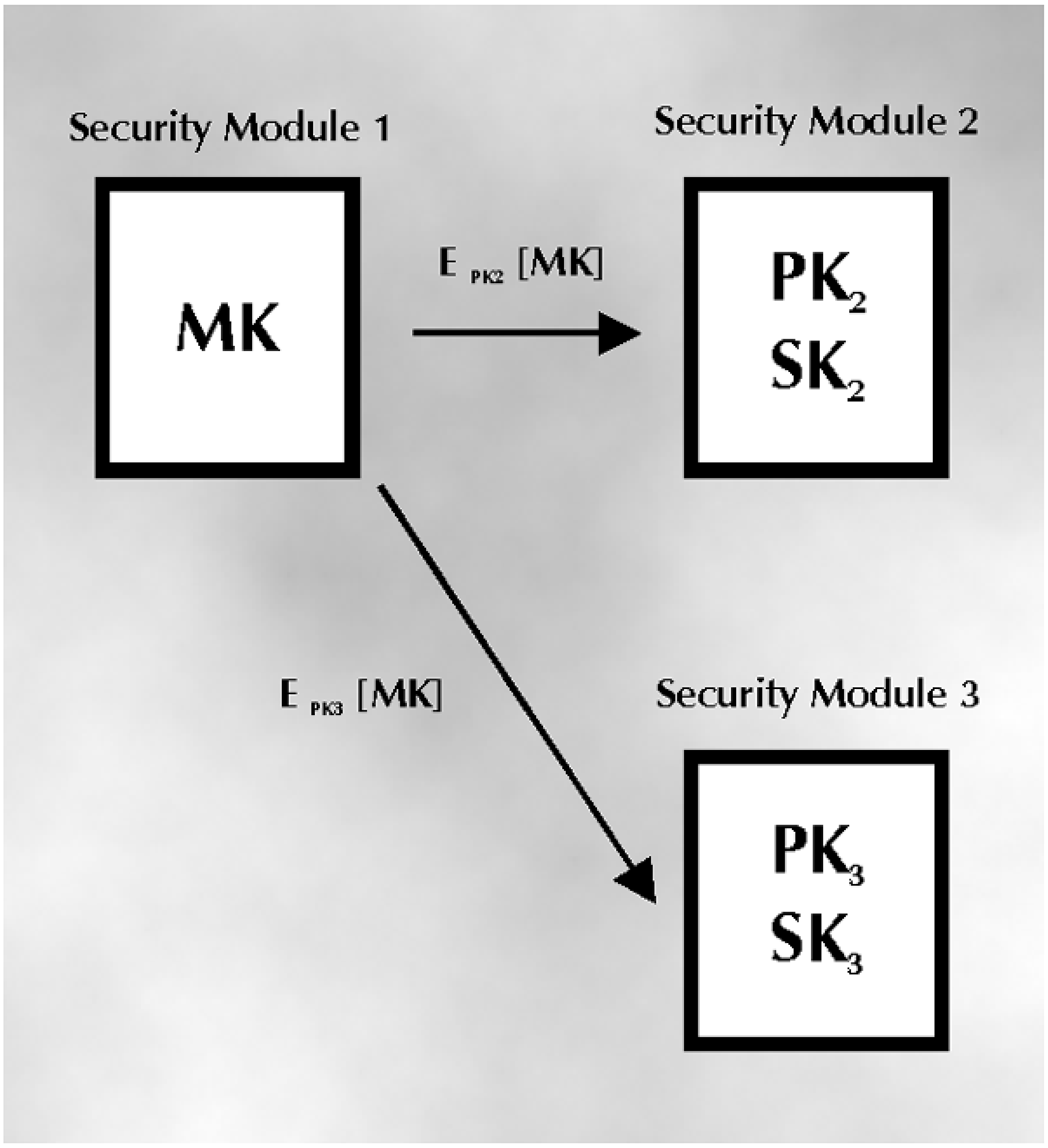


Figure 2: Master Key Establishment

Let us look at how this situation can be improved by using the RSA public key algorithm. We will consider a situation shown in fig. 2 where we will establish the source master key in three security modules. The security module 1 contains its own means to generate a random number which acts as the master key (MK). The security modules 2 and 3 are capable of generating their own RSA key pairs PK (Public Key) and SK (Secret Key). We should also note that this operation requires both modules to use an independently generated random number seed which is used to compute the keys. From previous discussions in the tutorial

we remember the RSA algorithm as follows,

- $C = M^e$ Mod N
- $M = C^d$ Mod N
- de = 1 Mod Lcm (p - 1)(q - 1)
- N = P * q

- where C = Cipher block
- M = Message block
- N = Modulus (entity public PK)
- e = encipherment key (Global public)
- d = decipherment key (secret key SK)
- P,q are prime numbers; Lcm is the lowest common multiple

using our simple number described previously for p,q of 5 and 11 we chose e to be a global constant of 3 from which d can be calculated as 7,

- 7. 3 = 1 Mod 20
- modulus N = p * q = 55

the point to be noticed here is that the security module computes the two primes p and q in a random fashion. The module then computes the secret key d (which never leaves the module) and the public modulus N.

The process of distributing the master key from module 1 to module 2 proceeds as follows,

a) The public key of module 2 (PK_2) is presented to module 1
b) Module 1 enciphers the master key with PK_2
c) The resultant cipher block is presented to module 2
d) Module 2 deciphers the block with SK_2 to recover MK

We can repeat the same operation so that all three modules now contain a copy of the master key generated by module 1. We can see here that the only vulnerability of this operation is in the presentation of the public key to the master security module. This is not a matter of secrecy but one of authenticity. Only the genuine public keys from modules 2 and 3 may be presented. This is a much simpler practical procedure than managing the secret keys themselves. We may also assume that Smart Cards may be used as the security modules since they can exhibit the necessary properties for a tamper resistant module.

Key Distribution

Having established the master keys in these security modules it now remains to set up the data keys in the total population of Smart Cards for the particular system under consideration. We can see that these data keys are derived from the master key (Fig. 1).

The problem now relates to the method of injecting the derived keys into the Smart Cards during the personization process. The master security module can easily calculate the derived key perhaps as a function of the Smart Card's serial number. However if this key is sent to the Smart Card across an unprotected channel then we must arrange comprehensive procedures to ensure this process is not compromised.

We can of course use similar operations to that described previously for establishing the master key, but one needs to be aware of the time overheads for the necessary public key operations. In practice the Smart Cards may not even be capable of implementing the cryptographic operations. A more practical procedure can be achieved by using a key encrypting key that is injected into the Smart Card chip earlier in the manufacturing process. The security module can be made capable of determining this key encrypting key and invoking the necessary operations. This considerably simplifies the procedural management of the personization process.

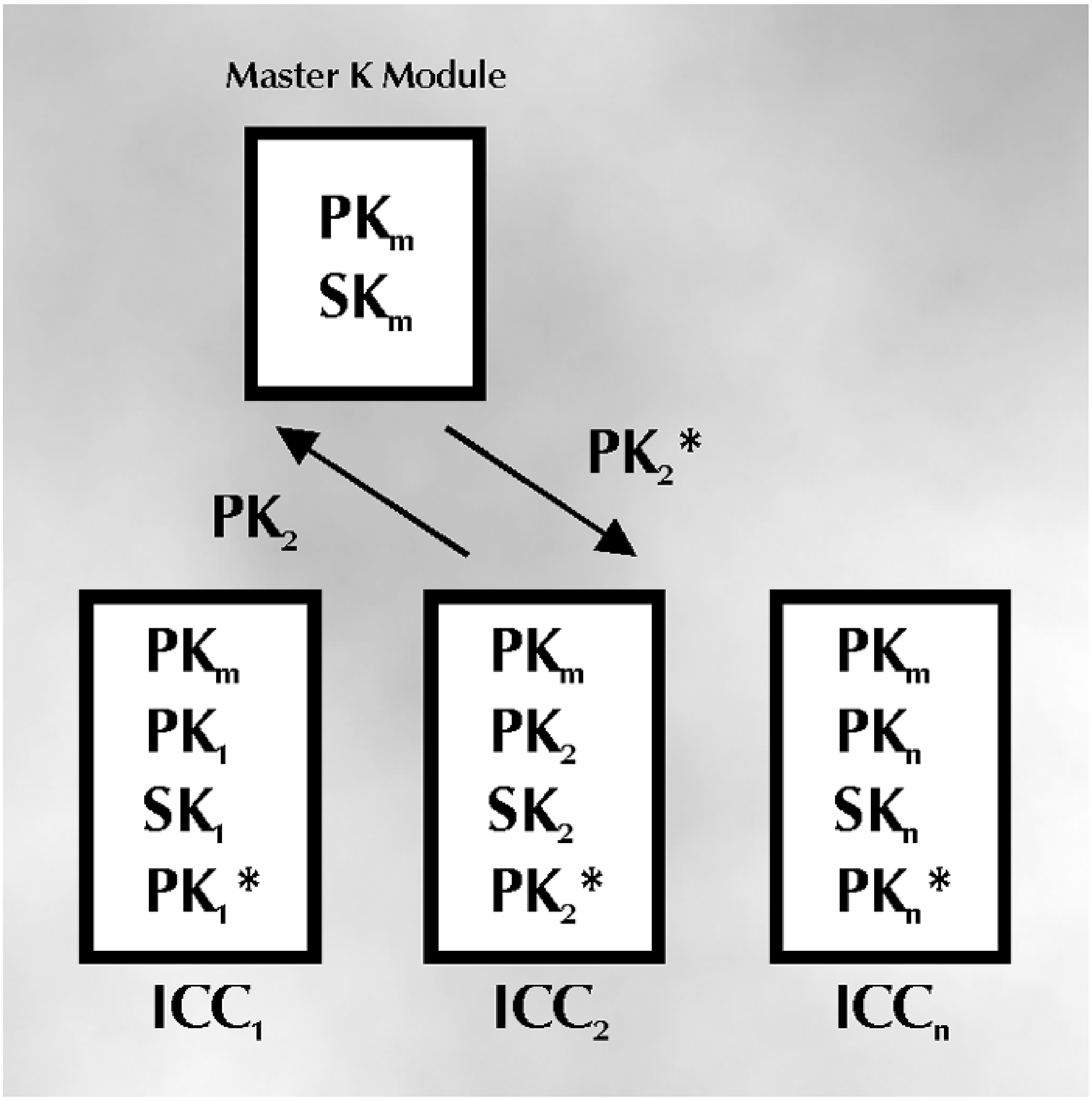


Figure 3: Public Key Establishment

A total public key system offers significant security advantages albeit at the expense of the complexity of the Smart Card chip. A new problem arises at the key establishment phase. In order for the Smart Cards or integrated circuit cards (ICCs) to interchange it is necessary for them to exchange their public key certificates. (fig. 3; * = certificate) When the cards are personalised these certificates need to be stored in the chip. If the ICCs calculate their own public key pair then the public key must be read from the chip and presented securely to the master key module for certification using its master secret key. This certification process is equivalent to the digital signature process. Again the problem relates to authenticity control not confidentiality. Thus the procedures must ensure that it is not possible to insert a bogus key for certification. A solution to this problem is for the ICC to use an additional key that will provide a necessary authenticity check. This key inserted earlier in the manufacturing process should only be known by the master key module. A similar effect can be achieved by injecting both the public and secret keys as well as the certificate after generation by the master key module.

In this case the confidentiality of the secret key must be preserved by using a pre established key relationship between the master module and the ICC. The public key of the master module can be used to check the key certificate and therefore the authenticity of the keys.

The main lesson to be gained from discussing these two approaches to key management relates to the operational security problem. In the case of secret key schemes such as DES it is necessary to preserve the confidential of the keys. Public key systems such as RSA by comparison require the authenticity of the keys to be ensured. An obvious extension to this principle is the need to be assured of the authenticity of the ICCs before invoking the secret key operations.

David Everett

Next month - multi-application Smart Cards

Smart Card Tutorial - Part 21

First Published in May 1994

Multi Application Smart Cards

Wherever you go the talk is all about multi application or multi function Smart cards. First of all we should have a go at defining the difference between multi function and multi application. Here I'm going to use the old fashioned computer terminology. Multi function referred to a tool that could implement different operations for a single client. Multi application referred to a tool that could implement different operations for different clients. If we stick to this terminology the commercial and security issues are readily apparent. The multi function card has a defined single owner who can set the commercial rules for use of the card and define the necessary branding issues. Most of the more complex applications to date have been along these lines in that the functions have all been defined by one provider or there has been a substantial cooperation to share the management of the Smart Card.

It is the multi application card that is really the more interesting. Here we have a Smart Card that is no more than an application carrier for the various application providers. These providers ideally should be totally unrelated. This raises a number of interesting questions,

a) Who owns the card ?
b) How is branding defined ?
c) What are the rules of application provision and operation ?
d) What happens when the card expires ?

These are just the surface of the commercial problem. One should not expect to see the general purpose multi application card take over the world just yet. We could carry on discussing these issues for some time but lets examine the technical issues more carefully. These issues can be summarised as follows,

1) How is the security segregation between applications enforced ?
2) How do applications authenticate the carrier ?
3) How does the carrier authenticate the application?
4) How are the co-residence rules enforced ?
5) How do you know what application exists on the card ?
6) How do you invoke the required application ?

The world of ICC Standards has really only addressed the last two of these questions with ISO 7816-4 (Inter Industry Commands for Interchange) and ISO 7816 - 5 (Numbering Systems and Registration Procedure for Application Identifiers).

These standards provide the following tools,

- An application identity registration scheme
- A directory file
- A file select command

We have mentioned previously the ability for application providers to register their applications. KTAS (Denmark) are providing the necessary administrative services although potential applicants should note that request for an application Identifier should be placed through their National Standards organisation.

The Application Identifier is made up of two parts,

- RID (Registered application provider identity)
- PIX (Proprietary application identifier extension)

It is of course the allocation of RIDs that will be undertaken by KTAS. Of greatest interest are applications intended for International use where the RID is defined to be of the form,

- Axxxxxxxxx (all elements hexadecimal - 4 bits each)
- (total 5 bytes or 10 hex digits)

The registered application provider may choose to add the optional PIX of up to 11 bytes (22 hex digits max)

The total application identifier (AID) is defined as follows,

AID	
RID (5 bytes)	PIX ($\leq$11 bytes)

This size of AID allows for significant scope in defining International Applications.

David Everett

Next month - Multi application Smart Cards continued.

Smart Card Tutorial - Part 22

First Published in June 1994

Multi Application Smart Cards - continued

Last month we established the concept of an application identifier (AID) for International use as defined by ISO 7816 - 5 (Numbering system and registration procedure for application identification). This was really the easy bit because the ISO standard allows an extensive number of options for application selection. Before we start digging in, it does not seem unreasonable to make a few assumptions on the environment in which the ICC (Integrated Circuit Card) and its application are going to be used. Lets consider a general model for the interaction with an application in an ICC (fig 1).

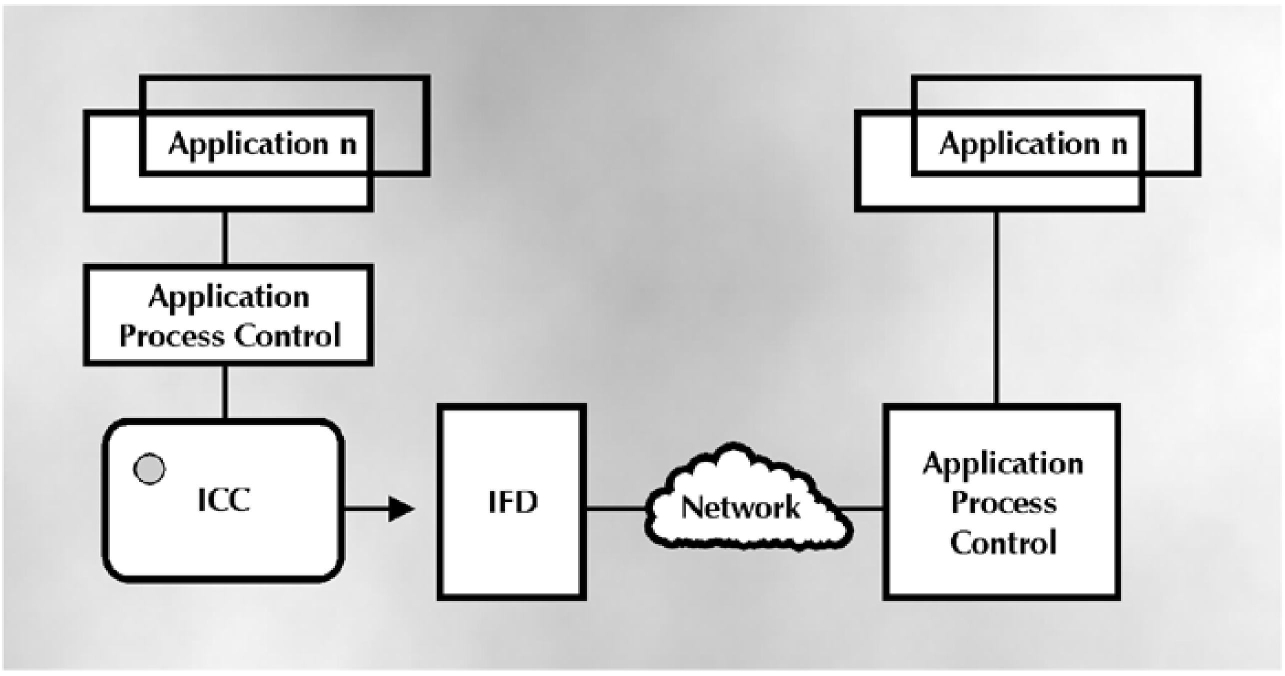


Figure 1: A Generalised Application Model

The ICC is considered to have one or more applications which may be selected by some application process control. The interface device (IFD) forms the basic interaction with the ICC in terms of ISO 7816-1,2,3 but is itself controlled by some application either resident in the IFD or remotely accessed across a network. Clearly the terminal system may also invoke one or more applications using a process control system by which means a particular application may be selected. Now here comes the first interesting commercial question - who selects what? intuitively one feels that the ICC should be the slave process and as such the choice of an application be made through the IFD. In particular it is obvious that whatever application is to be selected in the ICC a corresponding application in the IFD must control the interface to the ICC application.

According to ISO 7816 - 5 the identification of the application will enable the IFD,

- To ascertain that the card has the capability of initiating a particular application
- To identify the method by which a particular application in the ICC is evoked.

The standard determines three methods by which an application may be selected,

a) Direct selection using the AID
b) Indirect selection using information stored in the ICC either in a DIR file or an ATR file

c) Implicit selection where the application is considered to be invoked at the time of reset.

We may recall from previous discussions in the tutorial that ISO 7816 - 4 defines a file structure for the ICC and that the dedicated files (DFs) can contain application files. Thus to invoke an application we effectively need to select the relevant DF.

Let us look at each of these methods in turn,

Direct Selection

The assumption here is that the IFD wishes to invoke a particular application in the ICC and that it knows the AID for that application. In this situation the IFD will issue a 'Select File' command to the ICC where the AID is specified as the dedicated file name.

Clearly it is necessary that the IFD knows which application in the ICC to invoke and it proceeds by a trial and error approach. If the application (i.e the dedicated file) is present then the card will invoke the select file command and return a normal status response (90.00) to the command. If the application is not available then the response status bytes will indicate an error. These status bytes were discussed previously when exploring part 3 of the ISO 7816 standard. The select file command was discussed when we examined part 4 of the ISO 7816 standard.

Indirect Selection

The ICC may optionally contain a DIR or ATR file. Both of these files if present may contain application templates or the AIDs themselves. These data elements when present are always encoded using the ASN.1 (Abstract Syntax Notation) rules. The data fields are constructed as TLV (Tag, Length Value) objects, where the tag identifies the data semantics, the length represents the size of the data and value is the particular data element. The 7816-5 standard defines the following data elements encoded as shown in the table below,

TAG 1 byte	Length (L) 1 byte	VALUE (data element)	TYPE
'4F'	'01'to'10'	Application identifier (AID)	P
'50'	'00'to'10'	Application label	P
'51'	'00'to'7E'	Path	P
'52'	'04'to'7F'	Command to perform,see ISO/IEC 7816-4	P
'53'	'00'to'7F'	Discretionary data	P
'73'	'00'to'7F'	Discretionary ASN.1 objects	C
'61'	'03'to'7F'	Application template	C
TYPE: P = primitive ASN.1 object			
All other application class tags are reserved by ISO			

In its simplest form the DIR file could contain a set of records each of which contains the AID for each application present on the ICC. A typical record would be as follows,

Tag (1byte)	Length (1byte)	VALUE
4F $_{hex}$	10 $_{hex}$	AID (16 bytes$_{hex}$)

a more complex DIR record could define an application template which can contain a number of data elements referring to the application. By this means a typical DIR record might be as follows,

Tag (1byte)	Length	VALUE								
61 hex	2A hex	Tag	Length	value	Tag	Length	Value	Tag	Length	Value
		4F	10	AID	50	10	label	51	04	Path

In a normal commercial environment we could argue the case that the IFD should be configured to select a particular application as the result of some interaction with a user. It might reasonably expect that the ICC presented for use should contain that application and therefore the direct method of application selection would be the most appropriate. In many situations the alternative approach would be clearly wrong. If for instance a point of sale terminal were to examine the financial applications available on a presented ICC and choose the preferred application for the retailer then that would cause serious implications in terms of consumer rights.

Here the application template contains the AID, its path (in the file structure) and an application label.

The application label is a variable length (up to 16 bytes) data element that can be read by the IFD and displayed to the user as part of the man machine interface. It could be for instance an application brand name. The application template can be up to 127 bytes long.

In a practical situation the main advantage of the Indirect Selection method is the ability to check the ICC by reading say the DIR file to see if it contains the application that the IFD wishes to invoke. The ability to examine what other applications a particular ICC supports may in many situations be commercially debatable.

In any event both the DIR and ATR files are optional which means that the IFD cannot predict if this method of application selection is available. The application template may also contain one or more commands as defined in ISO 7816-4 that the IFD should invoke in order to select the application. The value of such techniques in a general purpose multi application environment is debatable.

Implicit Selection

The purpose of this form of selection is really to allow backwards compatibility with a single application ICC. After the ICC has been reset it is assumed that the application is already selected. The AID is contained in the historical bytes encoded by ASN.1 to confirm the particular application. As pointed out in the ISO 7816-5 standard this form of application selection is not recommended for multi - application cards.

In order to tie up a few loose ends we also need to expand on the AID. We referred last month to the first 4 bits of the first byte of the RID (Registered Application Provider Identity) as being 'A' for International registration. ISO 7816-5 defines 7 classes for the registration category values as follows,

'0'-'9'	As defined in IOS 7812
'A'	International registration
'B'	

	Reserved for ISO
'C'	Reserved for ISO
'D'	National registration
'E'	Reserved for ISO
'F'	Proprietary non-registered

Registration category 'D' is for registration by the National Standards Authority. The five bytes of the RID are defined as follows,

| D_{hex} CC1 | CC2 CC3 | SNA | SNA | SNA |

CC1 - CC3 are the country code BCD digits of the National registration authority. The remaining 6 BCD digits (SNA) are specified by the National authority. The PIX (Propriety Application Identification Identifier Extension) is applied as previously.

The registration category 0 - 9 is used to allow compatibility with IINs (Issuer Identification Numbers) as defined in ISO 7812. The first BCD nibble is the first digit of the IIN. Successive digits follow, where an odd number of digits is padded with F $_{hex}$. If a PIX is included as part of the AID it should be proceeded with FF $_{hex}$. The total AID must not exceed 16 bytes.

David Everett

Next month - Multi Application Smart Cards continued.

Smart Card Tutorial - Part 23

First Published in July 1994

Multi - Application Smart Cards - continued.

We have already discussed the ISO specifications that relate to the management of standards in an ICC. That however was only just the start of the story, now we need to try and understand what all this means in practice. Security as we shall see is an over riding issue in the management of Multi-application Smart Cards and one that as yet is not really covered by the ISO 7816 standard.

Lets start off by considering a software model for a Multi-application Smart Card as shown in figure 1. We have derived this model in so far as is possible using the ISO proposed file structure using a master file (MF), dedicated files (DFs) and elementary files (EFs). We could spend days discussing operating systems but their primary role is a resource management system. The purpose of the operating system is to select applications and to control the resources they need to achieve their functionality. The core of an operating system is the file structure which determines how applications can co-exist without interfering with each other.

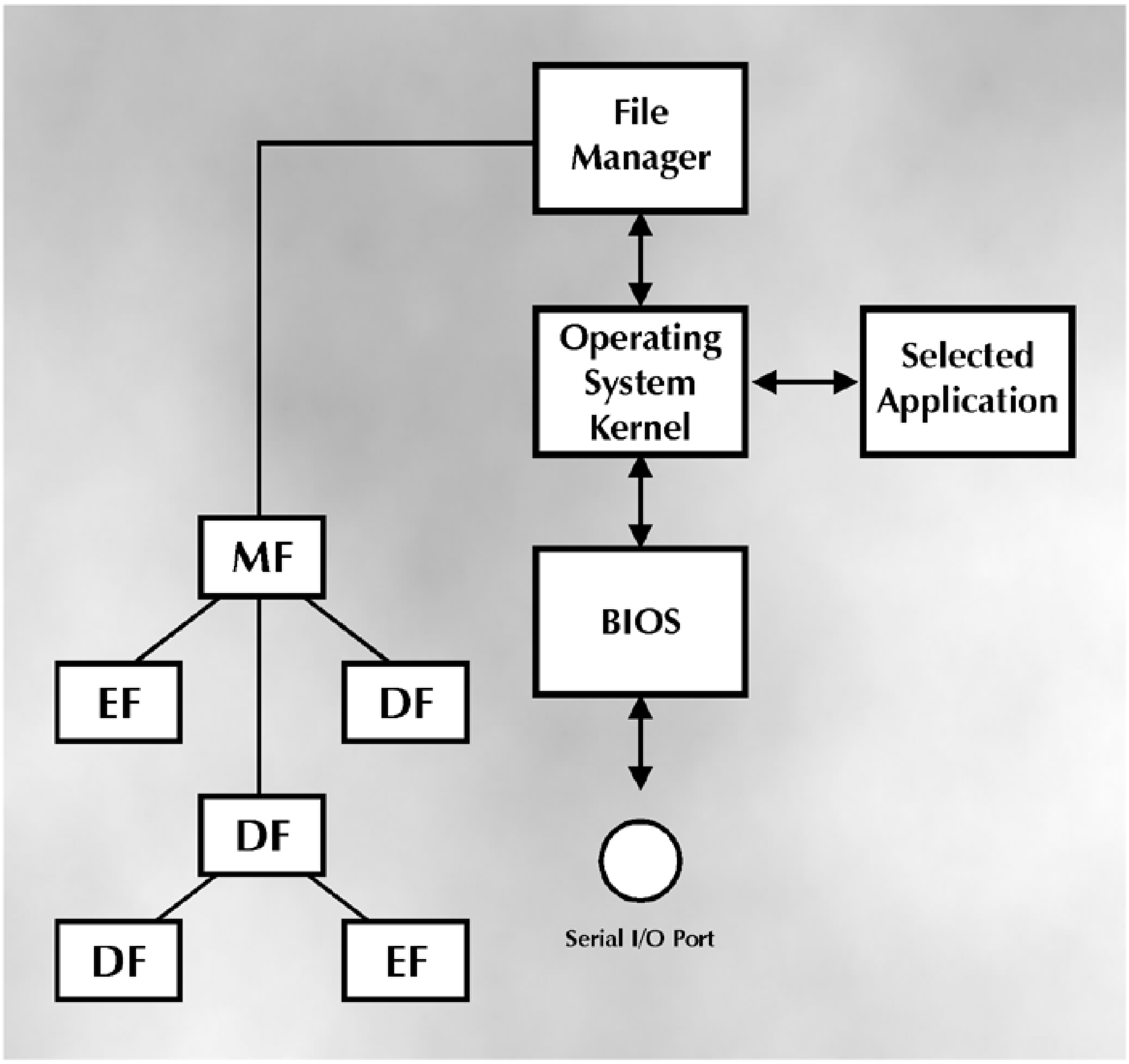


Figure 1: A Multi Application Architecture

There are four entities shown in our Multi-application architecture,

- Operating system kernel
- BIOS (Binary Input/Output System)
- File manager
- The application

The operating system provides the interfaces to the application and controls the two primary resources available on a Smart Card, the file manager and the BIOS (Binary Input/Output System) which is the driver that controls the serial interface available to the outside world.

The segregation between applications is totally dependent on the way the architecture is implemented. In figure 2 we show three ways by which the application can realise its functionality.

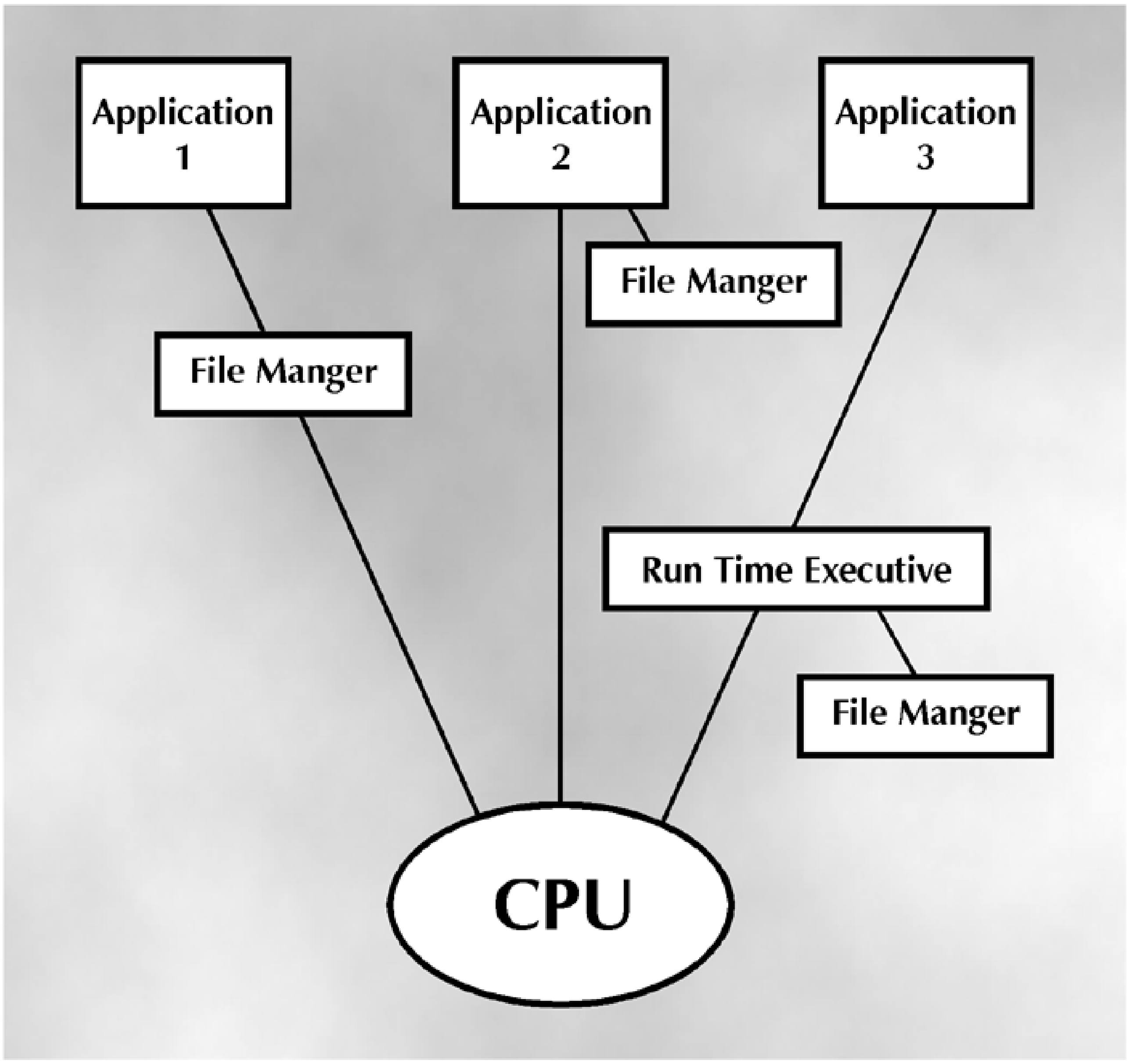


Figure 2: Application CPU and File Privileges

For application 1 the interface to the CPU is indirect since the application interface operates through the file manager. This is really the approach assumed by ISO7816-4. The task of the file manager is to control the access rights to the file structure. Before a file is made available for reading and writing the file manager can insist on proof of rights of access. This can be achieved by the use of a simple pass word or even more complex challenge and response techniques. These later techniques are widely used for authenticating a second party and are invoked by sending data to the other party which then forms a suitable cryptographic transformation such as a digital signature to produce a response that may be checked by the issuer of the challenge.

Once the appropriate authentication procedures have been successfully completed then the file manager allows the application access to the files dependent on the particular rights that have been established. This may be read only or read and write. Clearly at this stage the application has control of the data file and may invoke operations that manipulate the data.

If the application user is genuine as checked by the appropriate authentication techniques then we can assume that, subject to errors in the application, that the data will be manipulated correctly. Clearly the application will be restricted to those data files for which the user was able to prove the necessary access rights. The security of this approach is as good as the strength of the authentication technique. The use of a simple pass word may be inadequate if such a pass word can be determined by monitoring the Smart Card communications path or by delving into the terminal that invokes the application.

The approach adopted by application 2 as shown in figure 2 is the dangerous one. Here the application takes control of the CPU and whilst in an ideal world it should access the data files through the file manager this cannot be enforced. Any application that controls the CPU directly is clearly capable of bypassing any controls implemented by the file manager. Such applications are equally capable of controlling the serial I/O port directly avoiding any controls that might be implemented by the BIOS driver.

The application 2 type of system is in fact the one most commonly used today with micro-controller Smart Cards. If you are running a single application then clearly there is no problem but any extension to a Multi-application environment runs you into severe problems. In a Multi-function environment when a single commercial entity is responsible for a number of applications then this may not be a problem because they can take the necessary steps to ensure segregation of the application.

If two separate commercial entities want to share a single Smart Card for two different applications then we need to apply the appropriate validation and verification techniques. For anybody that has never tried this on a software program you are strongly advised to try and avoid the problem. It is extremely difficult to determine when a software module can accidentally or deliberately step outside its normal mode of operation. It is often practically impossible to exhaust the number of combinations of paths in the software that may be invoked depending on the various decision points that allow a change to the execution flow. In practice the only way you can handle this problem is by implementing a strict methodology in the software design cycle where you can show the correspondence of every line of code to a detailed requirements specification. This would require the two organisations wishing to co-habit the Smart Card to jointly develop the two applications.

It should be clear that this approach has a second limitation in that the whole life cycle of the application needs to be carefully controlled to ensure that the correct code modules are loaded on to the Smart Card and can not be subsequently changed. This in practice will require the use of software authentication codes and the necessary procedures to ensure that their checking is enforced.

It is actually the third approach as shown for application 3 in figure 2 that is the most interesting and is almost certainly the way forward for the future. Here the application interface to the file structure is controlled by a run time executive. The purpose of this run time executive is to ensure that the application cannot avoid the file manager. This technique can be achieved through software or by the use of memory control hardware in the Smart Card chip. The ST16xyz (SGS-Thompson) chip for example contains a memory access control matrix that is set when the chip is manufactured.

The purpose of this control matrix is to limit the memory that application programs are allowed to access. The access matrix is defined as follows,

Data area to be accessed →	RAM	ROM	EEPROM
Program contained in area ↓			
RAM	X	X	X
ROM	Access always allowed		
EEPROM	X	X	X

X = access allowed / denied can be set.

The use of this matrix is very powerful and forms an ideal base for Multi-application systems. In a normal system the operating system which includes the file manager will be implemented in the ROM (Mask Read Only Memory) area of the chip. The application would normally be loaded in the EEPROM (Electrically Erasable Programmable Read Only Memory) memory after the chip is fabricated. The RAM (Random Access Memory) area acts as the primary working memory for both the operating system and the applications. It is also possible to set up a program module to run in RAM space. Although this area on a Smart Card chip is very small, usually 256 bytes or less there may be occurrences when a designer wants to construct a self adaptive module. Whether or not such a technique is desirable we can pass on here for the moment but only to note that the ST16xyz matrix allows for memory control even in this situation.

From a practical point of view the normal matrix would probably prohibit a program in RAM by restricting access to RAM, ROM and EEPROM data space. A program contained in EEPROM would be allowed access to RAM data but would be denied access to EEPROM and ROM data. An application could still gain access to EEPROM data but this would have to be implemented through a application interface to the file manager running in the ROM program space.

The Phillips 83C852 crypto chip for Smart Cards has a simpler mechanism where programs running in EEPROM are prohibited from writing to EEPROM memory but they are still allowed to read such memory.

The run time executive could be implemented entirely in software. In this situation the application interface operates at an interpretive level where each instruction is actually executed by the run time executive directly. By this means the necessary memory partitions can be achieved. This method probably offers the greatest flexibility but is the more complex approach.

David B Everett

Next month - Multi-application Smart Cards continued.

Smart Card Tutorial-Part 24

First Published in August 1994

Multi-application cards continued:

Now the time has come to try and put everything together. We are going to invent a hypothetical multi-application card by which means we can examine the various technical and security issues that arise in practice. No attempt will be made here to justify the business case that may result in such a card being produced. In fact as we mentioned previously the commercial issues that surround the development of multi-application cards are complex. This includes not only branding issues but also the fundamental concerns surrounding liability and ownership and therefore the responsibilities of the various parties.

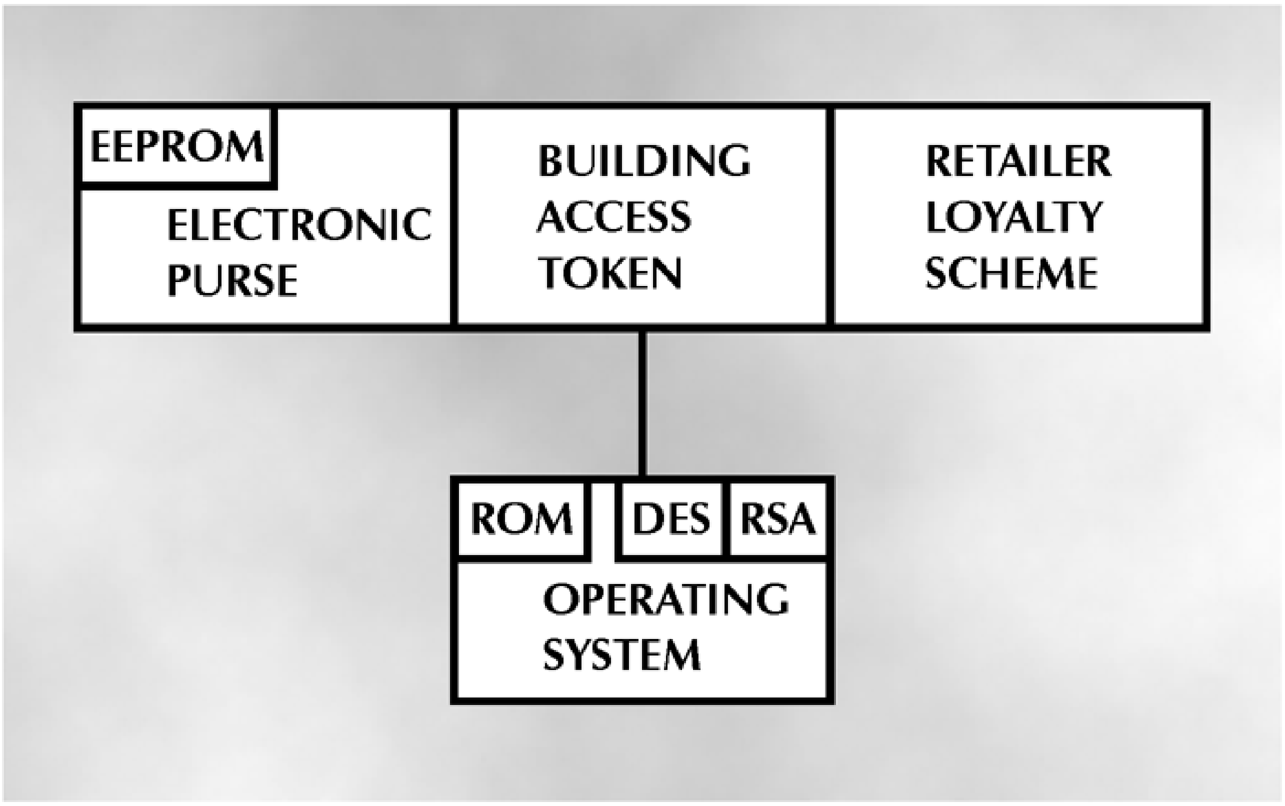


Figure 1: A Hypothetical Multi-Application Card

The structure of our card is shown fig. 1. The card contains an operating system in the ROM memory of the chip and three applications to be installed in the EEPROM memory. The applications have been deliberately chosen to be commercially independent with totally different business requirements. Lets examine the functionality of each of these applications.

The electronic purse application is provided by a bank and consists of a value store and a cryptographic protocol by which means payment can be made from the purse. We will assume that this protocol is based on the use of the RSA digital signature. The application will involve a number of commands issued to the card by the terminal that will invoke the necessary responses.

The building access application is designed to invoke a card holder authentication process by using a PIN and a signed response from the card to confirm the users identity. This involves the terminal challenging the card by a cryptographic process in this case using the DES algorithm.

The retailer loyalty scheme involves two processes, one that identifies the card holder and another function that stores loyalty points on the card that may be used at the relevant time to receive the appropriate award.

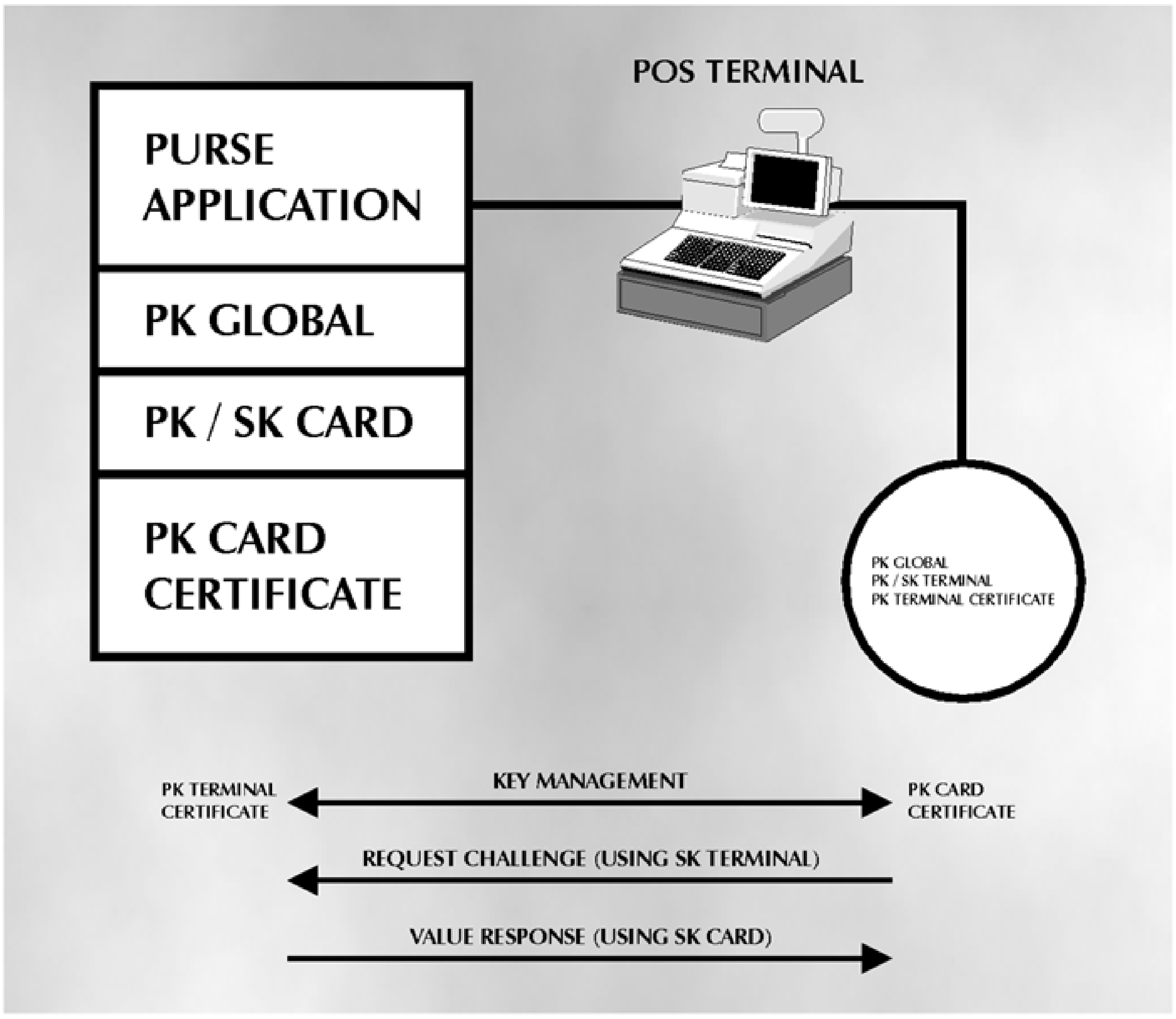


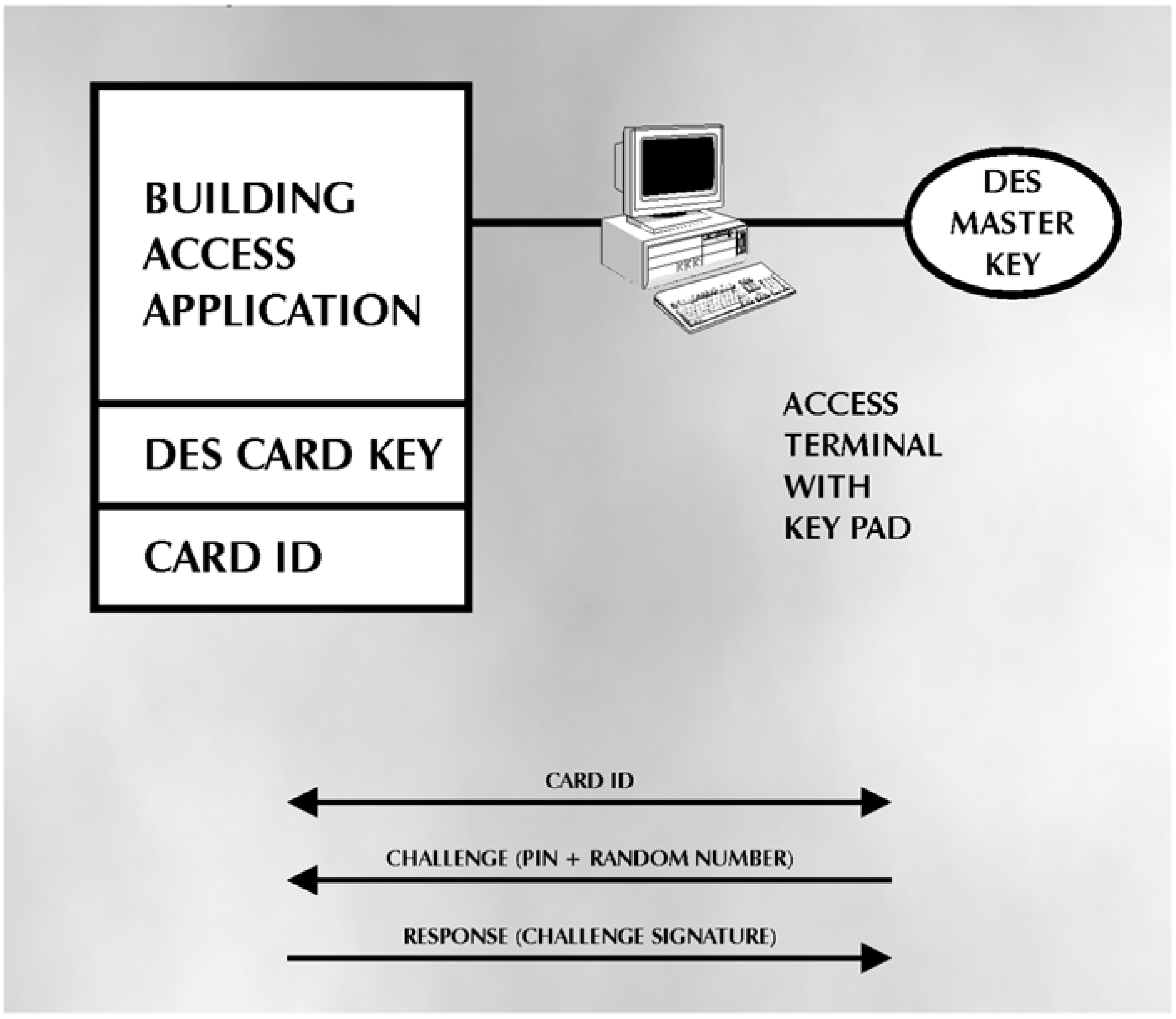
Figure 2a: The Electronic Purse Application

The mechanism for each of these applications is shown in fig. 2. They are shown in a very simplified format just to illustrate the principles underling the security architecture. In all cases we have assumed the availability of a security module in the terminal that can offer adequate protection for the keys and is capable of invoking necessary commands to the application in the card. In each of the applications described here a challenge/response mechanism is used with the appropriate random number or sequence number to prevent message replays. The necessary security services of authentication and data integrity are achieved by means of the cryptographic signature mechanisms.

The electronic purse application uses an RSA digital signature scheme where each card has a unique public key/secret key pair. The initial interaction with the terminal requires the exchange of public key certificates which have been generated by some higher authority or global key centre. In order to check these key certificates it is necessary to store an authentic version of the public key of this global key centre along with the relevant public key/ secret key pair and the necessary public key certificate.

The building access application is assumed to operate using the DES algorithm and has a derived key per card. This key is derived from a master key using the ID of the card.

The retailer loyalty scheme is also assumed to use the DES algorithm for protecting the points scheme but additionally has a file access key to prevent the users ID being read by a unauthorised agency. A derived unique key per card has been used based on the users ID and a points master key.



BUILDING ACCESS APPLICATION
DES CARD KEY
CARD ID
DES MASTER KEY
ACCESS TERMINAL WITH KEY PAD
CARD ID
CHALLENGE (PIN + RANDOM NUMBER)
RESPONSE (CHALLENGE SIGNATURE)

Figure 2b: The Building Access Application

Now let us look at how we can initialise the card to operate with these three applications. We have stated previously that the applications should maintain their own security which means that not only must each application support its own key management structure but that the operating system must ensure the necessary security segregation. Let us derive the necessary security requirements for the IC card,

1) The application must be able to trust the operating system
2) The operating system must be able to trust the applications.
3) Application security segregation must be assured
4) The cryptographic keys must be loaded securely

We can look at each of these motherhood statements to see what they mean in practice. The electronic purse scheme poses the greatest security requirement and clearly the provider of this application in particular needs to be assured that the operating system will protect the total working environment. It is likely that the operating system will be developed to the specification of the purse provider and will be subject to his certification process. Part of the purse application installation process will involve adequate steps to ensure the authenticity of the operating system. This may be achieved by procedural processes but is more likely to involve code authentication processes as well. We may also assume that in practice the cryptographic algorithms will be implemented in the ROM memory occupied by the operating systems. This is the most efficient use of memory space and allows the algorithms to be shared by the various applications.

However we have now run into our first problem, what is the process by which all the application providers can be adequately assured of the integrity of the operating system? If one of the providers is predominately responsible for the integrity of the operating system what is his liability to the other application providers in the event of a security flaw?

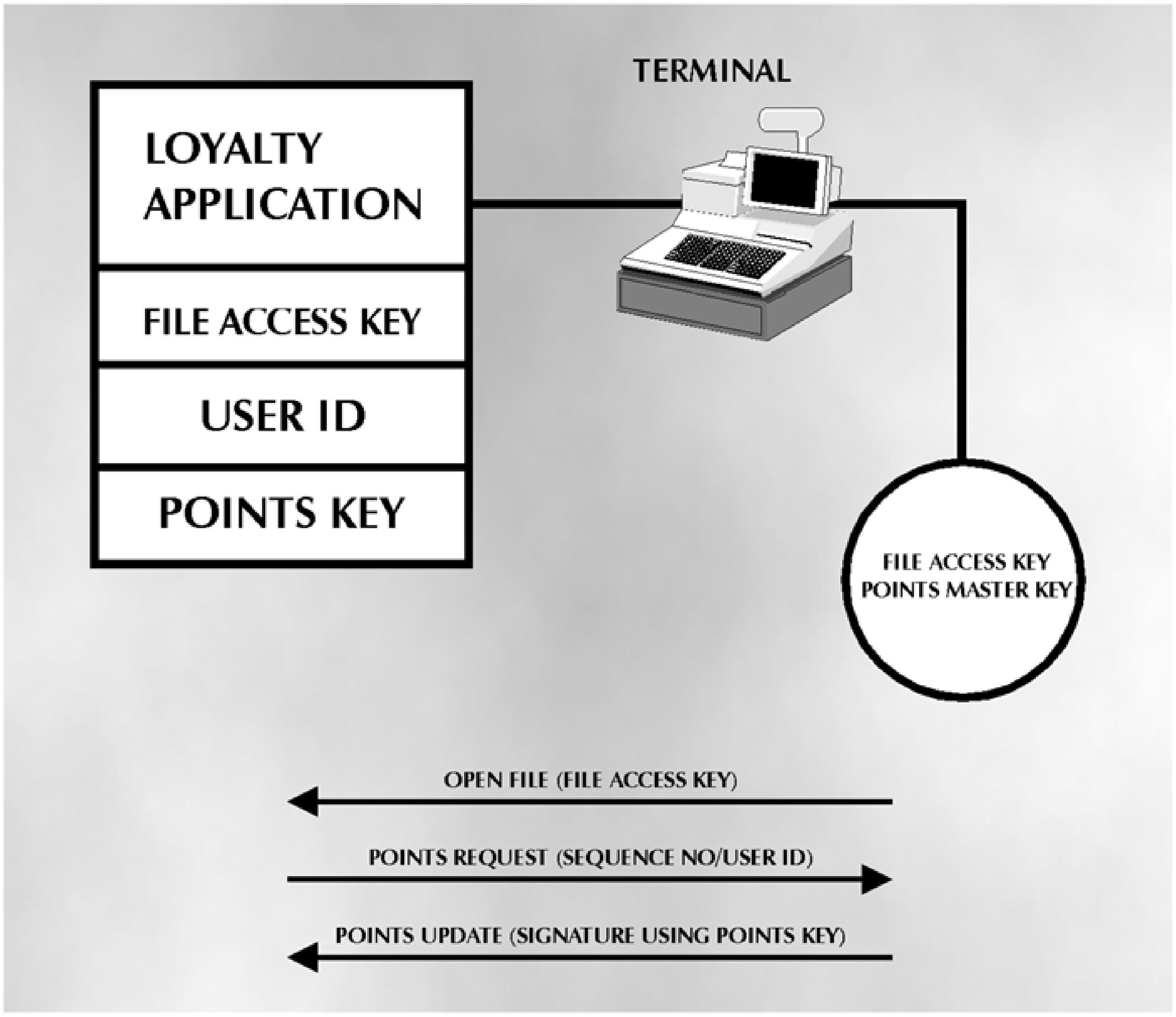

TERMINAL
LOYALTY APPLICATION
FILE ACCESS KEY
USER ID
POINTS KEY
FILE ACCESS KEY
POINTS MASTER KEY
OPEN FILE (FILE ACCESS KEY)
POINTS REQUEST (SEQUENCE NO/USER ID)
POINTS UPDATE (SIGNATURE USING POINTS KEY)

Figure 2c: Retailer Loyalty Scheme

The second requirement is necessary for the operating system to be able to ensure the necessary security segregation. We have previously mentioned various methods by which this can be enforced either by the hardware addressing structure of the chip or by the use of software interpreters. Conceptually the security segregation can also be achieved by certifying trusted applications and then checking the authenticity of the application when it is loaded by the operating system. For example the application could be signed using some digital signature function which is checked by the operating system before the application is loaded. It should be noted that this requires an additional key to be installed in the IC to check such signatures.

The application security segregation is totally dependent on the approach adopted to achieve the multi-application architecture. If the approach adopted is based on the operating system only loading certified application modules then additional procedures are necessary to ensure that such applications do not stray outside of their authorised boundaries. In practice this involves validation and verification of the software code modules and is very difficult to achieve even for relatively small applications. Enforced segregation by hardware logic in the chip or the interpreter approach is lightly to be far more effective and means that there is no need for the parties to certify individual applications. Clearly this is the easiest approach to pursue commercially.

The forth requirement relates to the difficulty of loading the cryptographic keys in a secure fashion. In any security system the installation of cryptographic keys is always a difficult practical problem. There is an additional problem with Smart Card applications that relates to the volume of cards that need to be initialised with the security keys. In many cases this is likely to be measured in millions. Whilst it is possible in principle to develop procedures by which such keys can be managed in practice most designers would base their architecture on the use of a key transport key. This key will need to be installed in the IC card in an early stage in the card life cycle. The various application cryptographic keys may be protected by this key to enable the key initialization process to be optimised with simpler procedure operations.

We have now created a new problem in that we have invented two new keys that need to be generated, operated and destroyed in a secure fashion. The application authentication key and the key transport key both relate to all the applications. The question that arises relates to who is in charge of these keys? Any compromise of these keys would effect all the applications equally. There are many different solutions to this problem but at the end of the day all the application providers will need to accept the common security process by which such keys are managed.

The purpose of this discussion was not to dismiss the concept of multi-application cards but just to point out the difficulties that need to be over come to make such cards commercially viable. There are solutions to all the problems described here but such solutions are very dependent on the business requirements established by the operators of the various card applications. The question of branding is entirely a commercial issue and one for which the solution may be the most elusive of all.

David B Everett
Next month - Contactless cards

Smart Card Tutorial - part 25

First Published in September 1994

Contactless Cards

In our previous discussions we have concentrated on the contact integrated circuit card (ICC) as defined by the ISO 7816 family of standards. Contactless cards have been around for some time but their penetration, certainly up to now, has been minimal. A number of emerging applications particularly in the mass transit area have been based on the contactless technology which leads to a wider interest in such technology. One can not help but notice the confusion and diffusion of the principles of such technology in the market place. In this part of the tutorial we will attempt to bring some perspective to the discussion where the advantages and disadvantages can be assessed in the light of any particular application.

In our thinking we tend to concentrate on the ubiquitous plastic credit card type of format more correctly defined as an ISO ID-1 card. The incorporation of an integrated circuit or chip into this card by the early entrepreneurs of Arimura in Japan and Moreno in France is now legendary. However the concept of small data carriers as radio tags predates their discoveries. These devices were used to identify objects by being tagged to them. When immersed in an RF (Radio Frequency) field they would respond with their relevant data field. One of the earliest applications and one that is seeing a major revival today is in the retail field where they can be used to prevent shoplifting or just to identify articles at a point of sale. Whether the economics of manufacturing such tags can extend to everyday goods purchased in the supermarket remains to be seen.

It is readily apparent that the technology of contactless communications is extremely well developed and is the core of our every day life. Of course the same is true for the technology surrounding data storage and processing. So when we are discussing contactless IC cards (CICC) what we are really looking at is bringing these technologies together in a reliable and low cost way into a physical object that meets our form requirements (eg ISO ID-1)

However there are as always a few other problems that in many ways form some of the major limitations. In the first instance we have to find a way of providing power in the card to run the chips, and we have to satisfy the appropriate regulators that any radiated signals will satisfy their current requirements in terms of frequency used and the transmitted power levels.

One of the objectives raised by the terminal manufacturing industry against contactless cards has been the lack of standards. There is an equivalent standard to ISO 7816 for contactless cards which is ISO 10536 (Identification Cards- Contactless Integrated Circuit Cards). Three parts have been identified,

- ISO 10536 - 1 Physical characteristics
- ISO 10536 - 2 Dimension and location of coupling areas
- ISO 10536 - 3 Electronic signals and mode switching

The first two parts are international standards whilst the third part is currently at the CD level (committee draft).

Contactless ICCs do not have the industry support that has been given to the contact card and this is clearly reflected in these emerging ISO 10536 standards. As of today there are only four major suppliers of contactless cards for general commercial applications.

There are a number of issues surrounding the use of contactless cards brought about largely by misunderstanding in the use of the technology. The advantages relating to the use of such cards in hostile environments are readily apparent, however the claimed advantages in performance and reliability are a function of much more than the contactless communications interface which we will expand upon further.

Contactless card technology

The components of the contactless card can be best considered by reference to the diagram shown in fig.1. The card can be broken down into the following functional components,

- Power supply
- Data communications
- Microcontroller

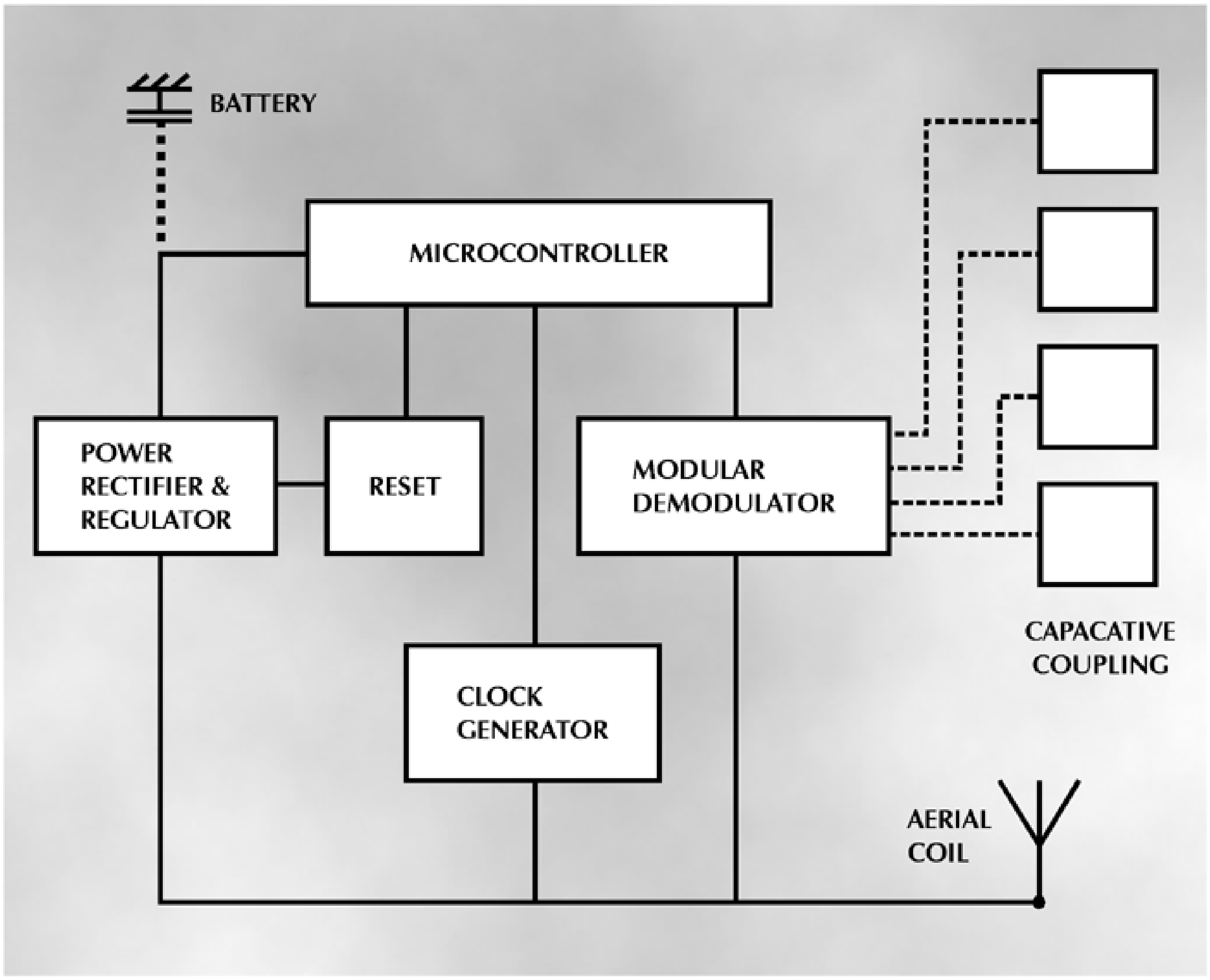


Figure 1: A Contactless Integrated Circuit Card

It should be noted that we have used a microcontroller here to compare with contact Smart Cards but it is understood that many contactless cards are based on memory only type devices. In other respects the principles are the same. It is the objective of the standards committee to have an alternative of ISO 7816 parts one to three where the higher parts can be considered common to all ICCs. We should note however even at this stage that only the AT&T contactless card has the ISO ID-1 card thickness of 0.8mm. The other major manufacturers cards are 1mm or more in thickness.

The supply of power to the IC chip can be by battery or by externally radiated power. Mitsubishi has developed a card using a paper thin lithium battery, this is new technology and most contactless cards obtain their power from an external source. This is achieved by absorbing power from an externally radiated source using a coil fabricated in a planar fashion on the card. The received signal is rectified and regulated to produce the necessary (typically 5 volt) voltage supply for the microcontroller.

Data can be communicated to the card by modulating the same carrier signal that provides the power source to the chip. There are in principle two common approaches to this problem, one is based on low frequencies (100KHz-200KHz) and the other on microwave frequencies (typically 2.45 GHz). Each approach has its various advantages and disadvantages. Low frequency systems require larger coils, have lower data rates and are more susceptible to ambient noise interference. However the lower frequency approach is easier to implement and is less likely to upset the regulatory authorities given the low level of transmitted energy. The allocation of microwave frequencies is more difficult. At the current time 2.45GHz has been allocated for wireless LAN applications or applications within a single building using spread spectrum technology. It would appear that 5.8GHz may become more widely adopted for point to point sources particularly for microwave links on motorway tolls.

The sinusoidala signal can be modulated by the digital data signal using one of three techniques,

■	Amplitude Shift Keying	(ASK)
■	Frequency Shift Keying	(FSK)
■	Phase Shift Keying	(PSK)

Amplitude shift keying is when the carrier is either on or off, frequency shift keying is when the carrier is switched from one value to another and phase shift keying is when the phase of the carrier is either in phase or out of phase. These modulation techniques are just digitised versions of the more general analogue modulation techniques of amplitude modulation (AM), frequency modulation (FM) and phase modulation (PM). Frequency modulation and phase modulation are closely related but in practice FM systems are more common.

The GEC card uses frequency shift keying for communication between the terminal and the card. The return channel is achieved by switching the loading of the aerial coil in the card so that more energy is adsorbed which produces an effect equivalent to amplitude modulation in the terminal circuitry.

As an alternative to modulating the power signal AT&T uses a capacitive coupling system to manage the data communication. This potentially gives a higher data communication rate but is also more fussy on correct alignment. The operating range for contactless Smart Cards is of the order of a few centimetres although RACOM quote up to 15cms under ideal conditions for their card.

Microcontroller

The microcontrollers used in contactless cards are usually based on standard designs such as the Motorola 6805. Such chips contain their own EEPROM memory and ROM memory. In this sense the security is little different to that achieved with a similar contact card. In both cases the communication path can be interrupted and it is necessary for the security designer to assume an insecure channel. The storage of sensitive data and the buses connecting them is still contained within a single chip and hence forms the necessary tamper resistant module.

The control logic for the microcontroller needs to provide the necessary reset circuitry and also the clock signals. The I/O line from the microcontroller is processed by the modulator /demodulator circuitry.

Main products

At the current time there are four major manufacturers providing main stream products.

The characteristics of these products are shown in the table below,

Manufacturer	Type	Power Transmission	Data Transmission	Data rate	Thickness
AT&T	Microcontroller	Microwave	Capacitive bi phase	19.2Kbps	0.8mm
GEC	Microcontroller	180KHz	FSK/ASK	9600 bps	1.3mm
Intag/Racom	memory	125 KHz	FSK/PSK (62.5KHz)	7812 bps	1.65mm
Mitsubishi/ Sony	Microcontroller	Battery	2.45GHz or 30MHz	9600bps	1.0/1.4mm

Perhaps all we need to comment on at this stage is the variation between the different products. Interoperability has a long way to go!

Contact/Contactless issues

There are a small number of features concerning cards that will dictate the suitability of one approach over another. We will compare these main features under the following headings,

- Reliability
- Performance
- Cost

Reliability

Reliability at the end of the day relates to the overall operation of the system. Availability is what it is all about. If we reduce our scope to just the card then we are comparing the reliability of the card itself and the reliability of the communications process. The comparison of the intrinsic reliability between a contact and a contactless card is straight forward. The contact card will always be more reliable barring abnormal wear of the connector plate because for an equivalent card there are less components and less connections. The reliability of the contactless communication channel compared with ohmic contacts is however another story. The main advantage of the contactless card is that the communication channel will operate in some hostile environments where contacts would be inappropriate. Clearly dirt or chemical type contamination environments are one such example but it should be noted that there are also environments where there is electrical noise that makes the contactless card inappropriate. In both cases the quality of the interface equipment is of paramount importance. In a similar fashion part of the environmental problems relate to vandal proof terminals and here the contactless interface is a significant advantage.

In either case for both contact and contactless cards the designer of the system should allow for a communications protocol that has the necessary levels of error detection and recovery.

Performance

Many people have been lured into the assumption that a contactless card is faster than a contact card. For an equivalent task this cannot be true since the processing time will be the same on the assumption of using the same microcontroller. The data transfer rate cannot be any different since the intrinsic limitation will be set by the data communication rate capability of the microcontroller for a particular clock speed. More to the point is the fact that the contactless card is limited by the ability to transfer adequate power to the microcontroller. It is generally believed that in practice this is likely to be limited to about 20mA. We note also however that the move towards portable battery equipment such as mobile phones is likely to have the same limits for contact cards.

There is no reason why contactless cards should be either more or less secure than a contact card. We would normally expect a contactless card to use a single microcontroller chip containing its own internal volatile and non volatile memory. For both types of card it is essential to assume that the communication path is insecure and therefore that the appropriate steps are taken where necessary. In some cases this may mean enciphering the data comms before transmission.

From simple engineering principles it is clear that the contactless card will be more expensive to manufacture. It contains more components and has a more complex fabrication process which involves the lamination of a complete card film substrate. A rule of thumb has often been used to propose that the contactless card will cost twice the price of a contact card to manufacture. However readers are warned that this is just one part of the overall system, printing and personalisation costs of the card as well as the cost for the infrastructure are important when determining the overall cost of the system. Price is very dependent on volumes and at the end of the day is highly correlated to market forces based on supply and demand.

Although the penetration of contactless cards to date has been very small in comparison to contact cards there is none the less a very real need for such technology in many applications. We have tried to put these advantages in perspective to show there is a future for both types of technology where the greater volumes are most likely to remain with the contact card.

David Everett

Next month - An update on Smart Card memories.

Smart Card Tutorial - part 26

First Published in October 1994

Smart Card memories

We often refer to the Smart Card chip as a device containing a Microcontroller and memory. This really belies the story behind some very sophisticated technology and in particular the developments of advanced memory techniques. This month we are going to look at the memory of Smart Card chips to explore the strengths and weaknesses of the various technologies.

In figure 1 we show the physical layout of typical but hypothetical chip as would be used in a Smart Card. Here we have identified two classes of memory,

- Volatile memory (e.g RAM)
- Non volatile memory (e.g ROM & EEPROM)

With static RAM the state of a memory cell is determined by the selling of a flip - flop. This is a transistor logic element that has two stable states either '0' or '1'. By the appropriate pulsing of the circuit can be alternated between the two states. Clearly when the power is removed there is no stable state so the memory is lost.

For each class of memory there are a number of different competing technologies with different characteristics. In particular we will consider the following parameters of the various technologies,

- Endurance (number of Readl'Write)

We have not separated cost because this is a function of the storage density and the fabrication technology.

Volatile Memory

This is generally referred to as RAM (Random Access Memory) of which there are two types,

- Static RAM (SRAM)
- Dynamic RAM (DRAM)

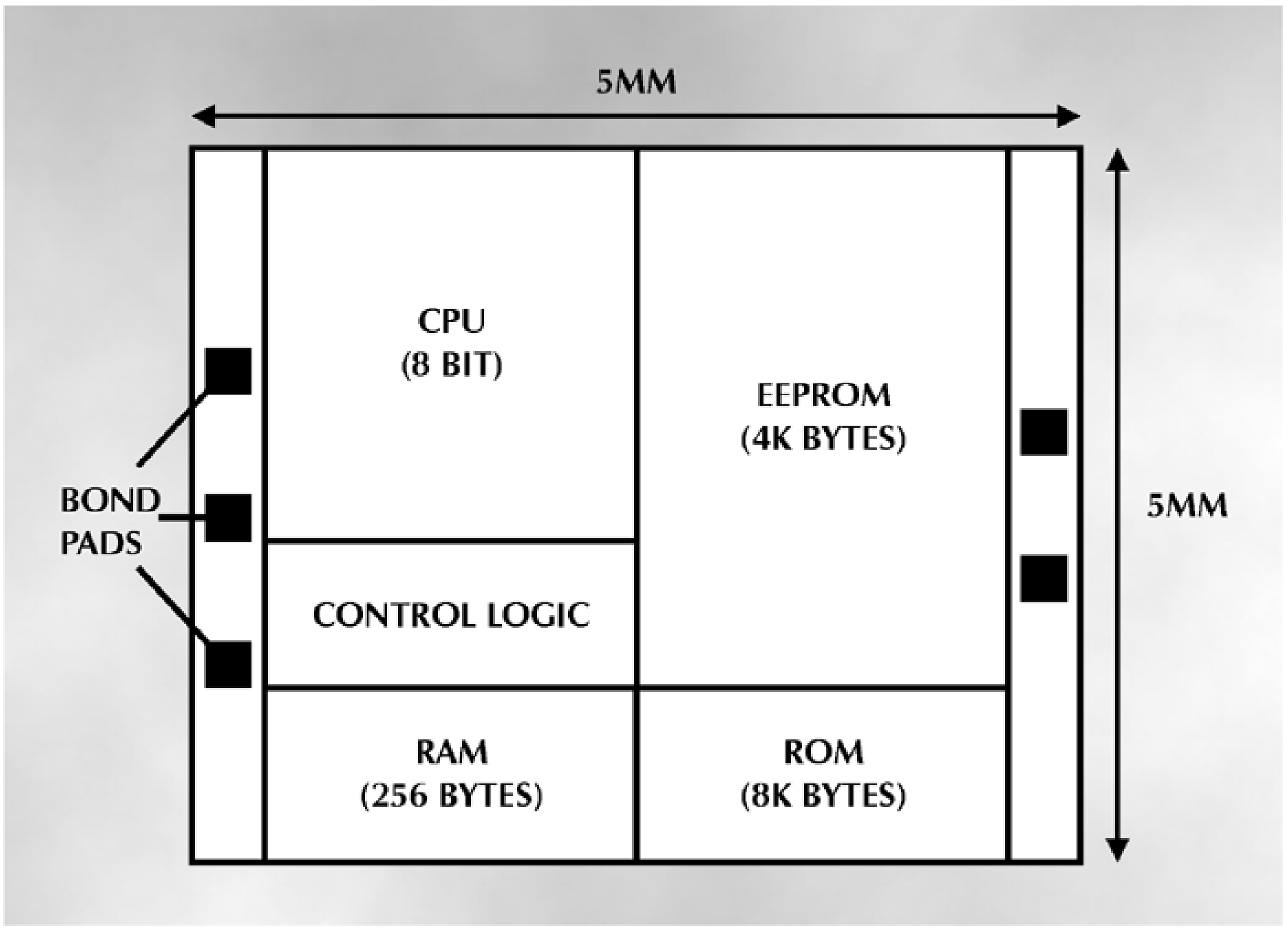


Figure 1: Chip Real Estate

In the case of dynamic RAM the memory state is determined by the storage of charge in a capacitive device. However this charge leaks away typically in less than a second and therefore needs to be constantly refreshed. This is accomplished by continually accessing all the memory cells in a background operation.

The principle advantage of DRAM is the ability to pack more memory per mm^2. But there is an enormous overhead in the refresh logic which only makes sense when developing large memory devices. Based on large memory chips (say 1M hit) there is a ratio of packing density of about 4:1 to the advantage of DRAM.

A typical Smart Card chip has a relatively small RAM area of 2K bits (256 bytes). At these levels of storage capacity the refresh circuitry would be an overwhelming burden. The static RAM is potentially faster than the dynamic RAM but for Smart Card devices running with clocks speed of 5 MHZ this would not normally be a consideration. All commercial Smart Card Microcontroller chips use static RAM. This also allows the sleep mode operation where the clock is stopped to save power whilst the chip is in a standby mode (e.g telephone SIM).

Non Volatile Memory

Well this is where the fun really begins. There are a large number of technologies where the words used to describe them can often be confusing. We will invent two categories,

- Non volatile read only memory (nv-ROM)
- Non volatile random access memory (nv-RAM)

The mask ROM used in the Smart Card microcontrollers is so called because the bit pattern is built in when the chip is made. The mask refers to the template by which the appropriate pattern is generated using photolithography techniques to manipulate the silicon surface. Older technologies were based on the use of a metallisation mask but most modern devices use an ion implantation process. The mask ROM memory is used to hold the primary program code for the Microcontroller. The data retention of these memory cells is good for the life of the chip. It is clear that this memory cannot be changed after manufacture and is therefore only capable of a read operation.

EPROM (Electrical Programable Read Only Memory) technology is still used in Smart Card chips and relates to a type of memory that can be written once but read many times. They are generally fabricated as MOSEETs with floating gates that can be charged (written) by avalanche injection. These gates are charged with electrons by a breakdown of the gate insulation layer using a high voltage (12 - 24v). This structure is sometimes referred to as FAMOS (Floating gate Avalanche injection Metal Oxide Semiconductor). Since the gate is not electrically accessible it can only be erased by exposing the cell to ultraviolet radiation which causes the stored charge to leak away by photoconduction. The EPROM memory cells have a data retention time that matches the life of the chip. Clearly in a Smart Card environment it would be normal to protect the chip against exposure to ultraviolet light. The normal packaging of the chip would provide this necessary protection.

The EPROM type of memory has been used for many applications such as telephone cards where it is not required to rewrite the memory contents. More recent EPROM technologies which have short channel lengths of 1 uM or less actually program the cell by hot electron injection as a result of impact ionization in the transistor drain region which occurs before the avalanche breakdown is reached.

Non volatile RAM

This is the area in which the greatest technology changes are taking place. It is also probably the most critical part of the Smart Card memory since it stores the dynamic application data. In applications such as the electronic purse the integrity of this memory is critical. It is also the memory that poses the greatest security vulnerability. The most general form of this memory is referred to as electrically erasable programmable read only memory (EEPROM).

There are three structures that are of particular interest,

- Metal-Nitride-Oxide-Silicon (MNOS - EEPROM)
- Floating-gate Tunnel-Oxide (FLOTOX-EEPROM)
- Ferroelectric Random Access memory (FRAM)

In an MNOS memory cell the electrons tunnel through a very thin silicon oxide layer (about 5nM) as the result of an applied electric field into the upper silicon nitride layer (about 50 nM) where they are trapped. The charge can be removed by applying a reverse electric field. This process is known as Fowler Nordheim tunnelling. The necessary electric field is generated by high voltage charge pump contained within the chip. The charge stored in the silicon nitride layer is very small, at about 50fC (approximately 300,000electrons) and is one of the reasons why you can't visually image an EEPROM memory cell using an electron microscope. This charge continues to leak (at a very low level) which sets a limit on the data retention time. The leakage is dependent on the thickness of the oxide layer but this also effects the tunnelling time (typically a few mS). The design process is usually optimised for a data retention time of ten years.

The FLOTOX process also results in the tunnelling of electrons across a some what thicker silicon oxide layer (about 10 nM) channel into a floating (surrounded by silicon dioxide) polysilicon gate. The Fowler Nordhiem process describes both the charge and discharge process. The intrinsic leakage process which effects the MNOS charge leakage does not apply to the FLOTOX process. However here the high electric fields applied in the charge and discharge process eventually cause defects in the thin tunnelling oxide which caused the charge to leak away.

There are advantages and disadvantages to both the MNOS and FLOTOX processes. MNOS is the older technology where the M for metal gate is today replaced by a polysilicon gate. The only commercial manufacturer of MNOS in microcontroliers is Hitachi. Most other manufacturers of microcontrollers use the FLOTOX process. In the early days the FLOTOX process was easier to fabricate resulting in higher yields and although this may still be true one suspects that improved fabrication technology will have reduced the difference.

The MNOS process should result in faster erase/write times because of the thinner tunnelling oxide and would also be more resistant to external radiation. The data retention time of FLOTOX should be higher for low levels of erase/write cycles.

In both cases the endurance of the memory is about 100,000 erase/ write cycles. There is no damage effect due to the read operation which just senses the stored charge.

Having gently grazed the surface of semiconductor physics we must look at the practicalities of using EEPROM memory. There are two stages involved in the writing of EEPROM memory cells,

- Erase
- Write

The erase cycle operates on all the bits in the byte, whilst the write cycle sets only those bits that are different to the erased state. Most microcontrollers allow a page to be erased and written at the same time. This page varies between the different manufacturers and is generally in the range of 4 - 32 bytes.

A programming cycle is accomplished by erasing the data block which for most manufacturers means taking the data memory to zero. Then the write cycles is actually a logical OR operation between the current state of the memory and the data to be programmed as is shown in the table below,

	Memory Data (Byte or page)	
Datatobeprogrammed	1011	0101
Initial EEPROM state	0001	0101
EEPROM after erase	0000	00 00
EEPROM after write (OR)	10 11	0101

With the particular values of data chosen here it would have been possible to leave out the erase cycle since the write operation (OR) on its own would have provided the correct operation. If you consider that the erase and write cycles typically take 5mS each then careful programming can significantly improve performance.

Most microcontrollers offer a combined erase/write step as a single operation in addition to the separate erase and write operation. The write operation can usually be applied to a part of the page whilst the erase and the combined erase/write must be applied to the whole page.

The word "FLASH" is now often heard when describing new memory devices. As applied to EEPROM it invariably refers to the architecture of the erase addressing. Here it is possible to erase large blocks of memory (perhaps the complete memory) in one operation. However the underlying physical process as described by Fowler Nordheim tunnelling still applies.

Ferroelectric Randon Access Memory
(FRAM ®)

(FRAM is the registered trademark of Ramtron International Corporation)

This is the newest of the non volatile memory technologies and the one hailed to replace EEPROM and FRAM (the core computer memory).

The ferroelectric effect, is the property of a particular material to retain electronic polarisation in the absence of an applied electric field. This stable bistate operation results from the internal alignment of dipoles within the ferroelectric material. The Perovskite crystal unit as shown in figure 2 is an example of a ferroelectric material. The construction of such memory cells is designed to make them insensitive to normally encountered external fields.

In figure 2 it is shown how the central atom moves between two stable states under the influence of an electric field depending on its direction. The atom remains at the same state even after the field is removed. The memory cell is constructed by sandwiching a thin film of ferroelectric between the transistor and metallisation layers of a CMOS process. The state of the cell is read by applying an electric field. If the atom needs to be moved (i.e to change the state) then the extra charge moved can be sensed. It should be noted that the cell will then need to be reset to its previous state. Such logic is built into the chip so that the operation is automatic.

The polarisation technique does not need the high electric fields necessary for the EEPROM tunnelling process and can be operated with the internal 5v system instead of the 12 - 15v required by the EEPROM process. The major difference however is in the performance of the read/write operation where the elecnic field needs only to be applied for lOOnS compared with the EEPROM's 5m5. A typical FRAM memory read/write cycle time is about 400n5. The endurance of the FRAM read/write cycle is also much higher for EEPROM with an endurance of 10^{10}cycles compared with $l0^{5}$. It should be noted however that the FRAM endurance must also take into account the read cycle which is non destructive in the case of EEPROM.

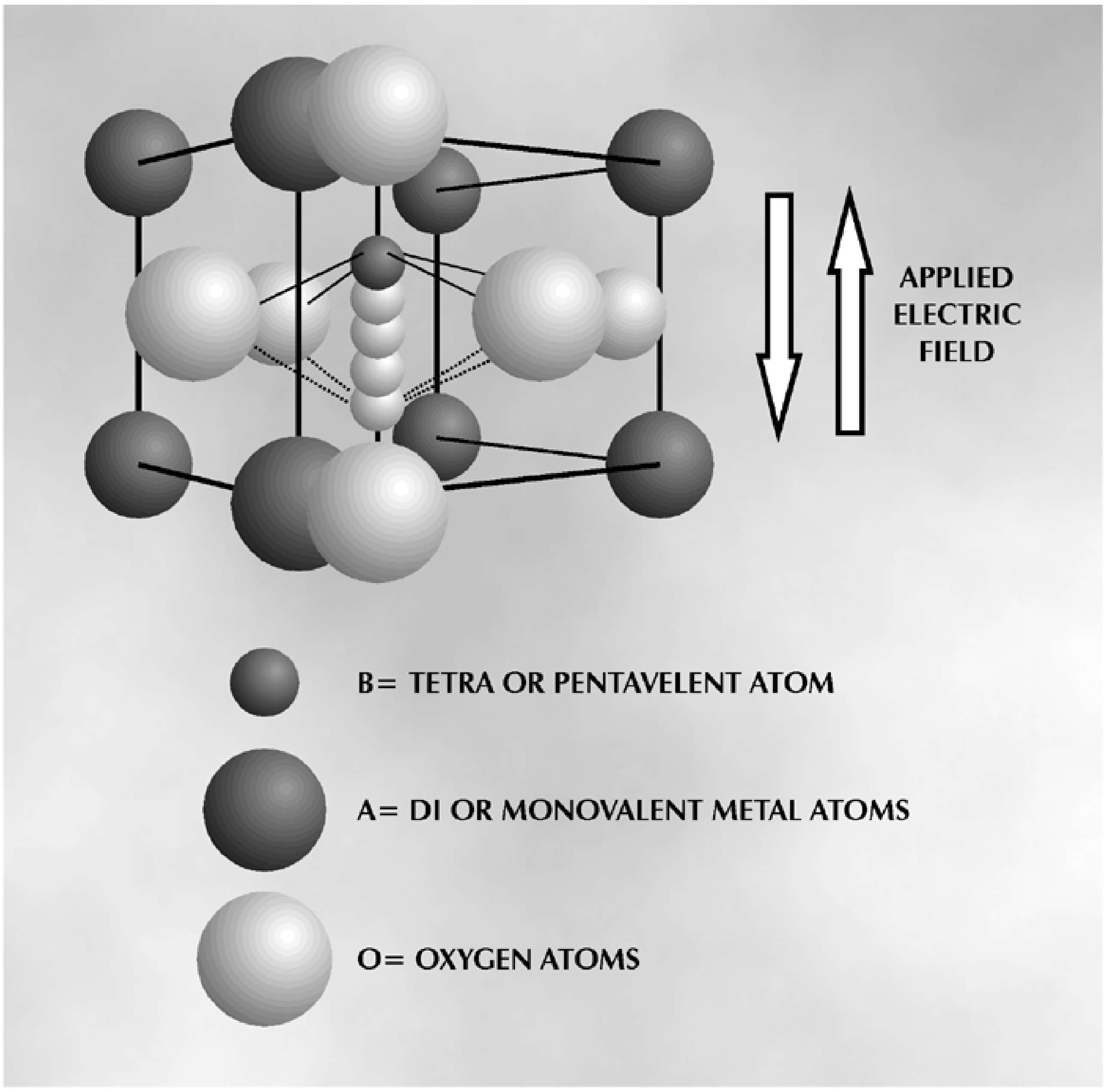


Figure 2: Perovskite Crystal Unit Cell

Comparison of memory technologies

	SRAM	ROM	EPROM	EEPROM	FRAM
Cell density (compared to SRAM)	I	X50	X30	X10	X5
READ time	10 - 150 nS	100 - 3()0nS	100 - 300nS	100 - 3OOnS	4OonS
Erase time	-	-	10 - 30min U.V	5-10 ms/page	-
Write time	10 - iSOnS	-	5 –50mS / byte	5 – l0mS/ page	400nS
Endurance					
Write/erase cycles	life	-	100(U.V)	10^4~10^6	10^{10}
Read/cycle	life	life	life	life	10^{10}
Data retention time	until power off	life	until erased	10 years	10 years

David B Everett