

Chip Programme – MAOS Platforms

MAOS Platforms

Technical Status Report

EUROPAY
I n t e r n a t i o n a l

Europay International S.A
198A Chaussée de Tervuren
1410 Waterloo, Belgium

Copyright © 2000, Europay International S.A.

This report is exclusively based on information received from vendors. Hence, Europay International does not warrant the accuracy or completeness of the information furnished herein.

Its purpose is only to analyse the available MAOS options and state of the art of technology, not to recommend a course of action to Europay International's members or to any other third party.

Control of Document

Author	Department	Author Role	First Created
Peter Johannes	SCD	Staff	6 Mar 98

Version History

Version No	Date	Change request	Description
00.001	6 Mar 98	-	Draft
00.002	8 Jun 98	-	Updated with comments on draft version
01.000	8 Dec 98	-	Updated with new data on the different systems and IC's and with the preliminary results of the hands-on exercises.
02.000	9 Nov 99	-	Yearly Update
	00		

Table of contents

1.	EXECUTIVE SUMMARY	1
2.	PROJECT OVERVIEW	3
3.	MAOS SYSTEMS	5
3.1	MULTI-FUNCTION SYSTEMS	5
3.2	MULTI-APPLICATION SYSTEMS	5
3.3	REQUIREMENTS FOR MAOS SYSTEMS	6
3.3.1	<i>Dynamic application management</i>	7
3.3.2	<i>Issuer involvement</i>	7
3.3.3	<i>Security</i>	7
3.3.4	<i>Model</i>	7
3.3.5	<i>Cost</i>	7
3.3.6	<i>Open vs. proprietary governance</i>	7
3.3.7	<i>Cross industry support</i>	7
3.3.8	<i>Availability</i>	7
3.3.9	<i>Performance and Development Environment</i>	7
4.	JAVA CARD	9
4.1	OVERVIEW	9
4.2	TECHNOLOGY ISSUES	10
4.2.1	<i>Virtual Machine</i>	10
4.2.2	<i>Security</i>	10
4.2.3	<i>Target requirements</i>	10
4.2.4	<i>Memory requirements</i>	11
4.2.5	<i>Hardware implementations</i>	11
4.3	PRODUCT ISSUES	11
4.3.1	<i>Specification</i>	11
4.3.2	<i>Platform</i>	12
4.3.3	<i>Development Tools</i>	12
4.3.4	<i>Certification</i>	12
4.3.5	<i>Usage</i>	12
4.4	SYSTEM ISSUES	12
4.4.1	<i>Java Card Forum</i>	13
4.4.2	<i>Application Loading</i>	13
4.5	SUMMARY - REQUIREMENTS MATCHING	13
4.5.1	<i>Dynamic application management</i>	13
4.5.2	<i>Issuer involvement</i>	14
4.5.3	<i>Security</i>	14
4.5.4	<i>Model</i>	14
4.5.5	<i>Cost</i>	14
4.5.6	<i>Open vs. proprietary governance</i>	14
4.5.7	<i>Cross industry support</i>	14
4.5.8	<i>Availability</i>	14
5.	WINDOWS FOR SMART CARDS	15

5.1	OVERVIEW	15
5.2	TECHNOLOGY ISSUES	16
5.2.1	<i>Virtual Machine</i>	16
5.2.2	<i>Security</i>	16
5.2.3	<i>Target requirements</i>	16
5.2.4	<i>Memory requirements</i>	16
5.2.5	<i>Hardware implementations</i>	16
5.3	PRODUCT ISSUES	16
5.3.1	<i>Specification</i>	16
5.3.2	<i>Platform</i>	17
5.3.3	<i>Development Tools</i>	17
5.3.4	<i>Certification</i>	17
5.3.5	<i>Usage</i>	17
5.4	SYSTEM ISSUES	17
5.5	SUMMARY - REQUIREMENTS MATCHING	18
5.5.1	<i>Dynamic application management</i>	18
5.5.2	<i>Issuer involvement</i>	18
5.5.3	<i>Security</i>	18
5.5.4	<i>Model</i>	18
5.5.5	<i>Cost</i>	18
5.5.6	<i>Open vs. proprietary governance</i>	18
5.5.7	<i>Cross industry support</i>	18
5.5.8	<i>Availability</i>	18
6.	MULTOS	19
6.1	OVERVIEW	19
6.2	TECHNOLOGY ISSUES	19
6.2.1	<i>Virtual Machine</i>	19
6.2.2	<i>Security</i>	19
6.2.3	<i>Target requirements</i>	19
6.2.4	<i>Memory requirements</i>	19
6.2.5	<i>Hardware implementations</i>	20
6.3	PRODUCT ISSUES	20
6.3.1	<i>Specification</i>	20
6.3.2	<i>Platform</i>	20
6.3.3	<i>Development Tools</i>	20
6.3.4	<i>Certification</i>	20
6.3.5	<i>Usage</i>	21
6.3.6	<i>System Issues</i>	21
6.3.7	<i>MAOSCO</i>	21
6.4	SUMMARY - REQUIREMENTS MATCHING	21
6.4.1	<i>Dynamic application management</i>	21
6.4.2	<i>Issuer involvement</i>	21
6.4.3	<i>Security</i>	21
6.4.4	<i>Model</i>	21
6.4.5	<i>Cost</i>	22
6.4.6	<i>Open vs. proprietary governance</i>	22
6.4.7	<i>Cross industry support</i>	22
6.4.8	<i>Availability</i>	22
7.	MATCHING THE REQUIREMENTS	23
8.	BENCHMARK	25
8.1	DEVELOPMENT ENVIRONMENTS AND COMPONENTS USED	25
8.1.1	<i>Native platform</i>	25

<i>MULTOS</i>	25
8.1.3 <i>Java Card</i>	25
8.1.4 <i>Windows for Smart Card</i>	26
8.2 NATIVE DEVELOPMENT.....	26
8.2.1 <i>Keil environment</i>	26
MULTOS DEVELOPMENT	27
8.3.1 <i>Development in MEL</i>	27
8.3.2 <i>MDS environment</i>	27
8.3.3 <i>Hitachi development environment</i>	28
8.3.4 <i>SwiftCard development environment</i>	28
<i>MULTOS Application Loader</i>	29
8.4 JAVA CARD DEVELOPMENT	29
8.4.1 <i>Development in Java</i>	29
8.4.2 <i>G&D Sm@rtCafé environment</i>	30
8.4.3 <i>GemXpresso RADII 211 environment</i>	30
8.5 WINDOWS FOR SMART CARDS DEVELOPMENT.....	31
8.5.1 <i>Development in Visual Basic</i>	31
8.5.2 <i>Microsoft Visual Basic 6.0 environment</i>	31
8.6 COMPARISONS BETWEEN ENVIRONMENTS	32
8.6.1 <i>Comparison between Sm@rtCafé and MULTOS MDS environments</i>	32
8.6.2 <i>Comparison between Keil C and SwiftCard C development environments</i>	32
8.6.3 <i>Comparison between Sm@rtCafé and SwiftCard C development environments</i>	32
8.6.4 <i>Comparison between Sm@rtCafé and GemXpresso RADII 211 development environments</i>	32
8.7 IMPLEMENTATION DATA.....	33
8.7.1 <i>Code Size</i>	33
8.7.2 <i>Application Validation</i>	33
8.7.3 <i>Transaction Timings</i>	33
8.7.4 <i>Command timings</i>	35
8.8 ISO AND EMV COMPLIANCE ISSUES.....	38
<i>MULTOS</i>	38
8.8.2 <i>Java Card</i>	38
8.9 CONCLUSION	38

1. Executive Summary

Today's multi-function platforms are based on platform specific executable code (GeldKarte, Chipper, MFC). The security of these platforms requires verification of the executable code of all applications by the issuer. Application code is target platform and target processor dependent. The business requirements of openness and portability are not met, or are met on a per issuer basis.

Multi-application platforms are based on platform independent interpreted code (MULTOS, Java Card). Security of the platform is guaranteed by the design of the platform and is not dependent on the application. Therefore the issuer does not need to inspect the code of each application that is loaded on the card. Application code is independent of the target platform. Only interpreter based technologies meet the business requirements of portability and openness at this moment.

A wide range of platforms and suitable IC's will be available in the near future from different vendors. The initial target card price will be around 6 \$ for a 16K EEPROM component in 1 million volume.

All vendors are concentrating on the card platform. This however presents an answer to only part of the problem. Currently there are no back office and terminal infrastructure solutions available for loading applications in the field, although several vendors are starting with beta testing their products in more or less large pilots.

2. Project Overview

In October '96 the Europay board approved a 4 phase process to analyse the MAOS options and to recommend a course of action protecting the strategic interest of Europay members. This document reports on the second phase i.e. the analysis of the options available and the state of the art of the technology. It represents the gathering of information from the different vendors.

Meetings and discussions with the following vendors have been organised and / or questionnaires have been sent to: Bull, De La Rue, Gemplus, Hitachi, IBM, ING, Java Card Forum, SUN Microsystems, MAOSCO, Microsoft, Mondex, Motorola, Nec, Schlumberger, SGS-Thomson, Siemens and Toshiba.

Significant delay was encountered due to difficulties with NDA's and due to the fact that the technology is still in the early stages of development. Therefore hard data was difficult to obtain.

Since last year, the different platforms have reached a certain maturity and have become more easily available. Significant effort was devoted to obtain comparative measures to evaluate the different platforms.

It should be noted that this report focuses on the card platform and does not address system issues such as the application loading infrastructure, impact on the network and back office.

3. MAOS Systems

MAOS systems historically were developed in 2 variants, Multi-Function systems and Multi-Application systems. The main difference lies in the issuer involvement in the operation of the system and the extent to which they can be tuned to meet issuer requirements.

3.1 Multi-function systems

Multi-function systems allow multiple applications on the same platform without the guarantee that the applications do not interfere with one another. The system therefore cannot guarantee that data that is private to one application remains inaccessible to another application.

To guarantee waterproofing, the issuer needs to inspect the code of each of the applications that is loaded on the card. In addition, since executable code is used, the exact configuration of the card needs to be known to be able to link application code with libraries that reside on the platform. Therefore an important back office is needed for dynamic loading of applications.

The application executable code is platform dependent. A single application that is to be loaded on several different platforms needs to be developed several times.

Almost all systems with multiple applications on the same platform fall at present in this category.

In general these systems offer better performance since they are implemented in executable (native) code and they do not need to perform runtime security checks.

3.2 Multi-application systems

A multi-application operating system is an operating system that allows multiple applications to reside on the same platform with the guarantee that applications do not interfere with one another. This means that data that are private to one application remain private and other applications can not read or modify these data.

Basically there are two ways to provide this application independence or waterproofing:

- **Interpreter:** The application code is compiled into byte code and afterwards interpreted on the platform. The interpreter fetches each byte code one by one and checks if the access conditions to the resources can be granted.

- **Memory manager:** The memory manager inspects each address and checks if the address falls in the region that was allocated to the application. Access to addresses outside this region is blocked.

Interpreter based operating systems allow an abstraction of the underlying hardware platform. This results in code which is platform independent. The exact configuration of the platform is not needed since typically linking can be done at execution time. Dynamic loading of applications is easy since less information is needed to describe the status of the card.

The applications only need to be developed once since the interpreter abstracts the application code from the underlying hardware.

The applications resident on a card don't impact the way of programming other applications which could be loaded. Applications on the card are totally independent of others.

A MAOS system allows to dynamically add new applications while the card is in the field and to tailor the card to the needs of each of its customers individually. Depending on the business model, this task is managed by an issuer, a scheme, a service provider, ...

A neutral third party can perform the key management such that the card issuer can issue load certificates to application providers without having to inspect the code.

Given the underlying hardware requirements, and the improved functionality these systems are the platforms of the future.

	Multi-Function Systems	Multi-Application Systems
Waterproofing	Code inspection by issuer	Guaranteed by the system
Application code	Platform dependent	platform independent
Runtime engine	Executable	Interpreted / Mem. Mgr
Application development	Platform dependent	Platform independent

Table 1: Comparison between multi-function and multi-application systems.

In the remainder of this report and also in the next agenda item, the benchmarking of different applications, we only consider Multi-Application systems, as they are the only platforms that can be discussed objectively. Indeed, the Multi-Function platforms are always tuned to a given issuer, so the requirements it meets are the ones the issuer is prepared to fund and is satisfied with.

3.3 Requirements for MAOS systems

To bring a better understanding of the issues involved in selecting a MAOS system, it is necessary to define a number of criteria. The criteria we judge meaningful are listed and explained below.

3.3.1 Dynamic application management

With dynamic application management, we refer to the ability of the platform to allow the issuer of the card to update the card once it is in the field. Is there support for the secure loading / execution / locking / unlocking / deleting of card applications / software? For administrative tasks?

3.3.2 Issuer involvement

How closely does the issuer need to be involved in the development or certification of applications that are going to be downloaded. Does he need to scrutinise each application or can he rely on the system to protect himself from program errors or malicious attacks?

3.3.3 Security

What security rating is achieved? Is it important? What does it cover? Who grants security approval?

3.3.4 Model

Does the MAOS allow the issuer to be in full control of his card? Can he delegate the operation to a third party? What is the extent to which the cardholder can autonomously pick applications?

3.3.5 Cost

What is the cost of a MAOS platform? Which are the additional costs for loading, deleting and in general managing the applications? Are there competitive licensing terms?

3.3.6 Open vs. proprietary governance

Is there an industry wide body that owns and manages the specifications or is it a company initiative? Do banks have a say? Is the platform open for future evolution?

3.3.7 Cross industry support

Is there a support for the platform outside the banking industry? More specifically, in the telecommunications sector?

3.3.8 Availability

Can you order samples? Can you order production quality cards in large quantities? Is there any significant acceptance in the market? Is there a certain independence of hardware platforms?

3.3.9 Performance and Development Environment

This is such a changing and volatile topic that it is included as a separate document giving the current state of affairs.

4. Java Card

4.1 Overview

Java is a high level object-oriented programming language which was proposed by SUN Microsystems. It resembles C++ but is much less complicated while retaining the most important language features. Java is object-oriented (with single inheritance), statically typed, multithreaded, dynamically linked and has automatic garbage collection. Application developers can use a number of Application Programmers Interfaces (API) when developing new applications. These are standard interfaces to packages that contain a number of utilities like I/O, network, GUI, and application domain specific functions. The Java API framework is open and extensible. Specifications for each interface are developed by industry-wide specialists in each area. The APIs are published and open for industry review. The large set of available APIs makes the life of the application programmer easy since he can (re)use a lot of standard code. Use in commercial products is subject to licensing.

The Java source code is compiled into Java byte codes which are executed on the Java Virtual Machine (JVM). This is a one stack 32 bit VM which recognizes 201 instructions. The Java byte code compiler compiles the source code into class files. These files contain the byte code and all other information needed to execute the code.

The Java Card VM and language specifications are a subset of the original Java VM and language specifications. It was adapted due to performance and resource issues of the implementation of the VM in a very constrained and minimal environment. A number of data types and language primitives from the standard JVM are not implemented. Since the language and byte code are a subset of the standard language and byte code, all the traditional compilers and development environments can be used. After compilation the class file is further compressed for the reduced environment of the card by a post processor. Up to recently, the various manufacturers implemented different post-processing schemes. SUN Microsystems has put an end to that by releasing a single specification for a common CAP format at the Java Card Forum meeting, dd. 3-4 December 98. This specification is now part of the 2.1 release, which was made available on 24/2/1999. Also part of this release are the definition of two standard classes, the I/O class and the cryptographic services class. The development kit including the converter and spec updates was posted in Nov. 99.

4.2 Technology Issues

4.2.1 Virtual Machine

The Java Card VM recognizes 187 instructions and has been derived from the standard JVM. The stack size is not specified (implementation decision). The byte codes that it recognizes are a subset from the standard JVM. In addition the functionality of the standard Java VM has been broken into different pieces of which only a small portion is implemented on the card. The rest of the services such as the byte code verifier is implemented outside of the card.

Due to the object-oriented nature of the VM, the code density of the standard JVM is not suited to be implemented on a card. Therefore the standard byte code is compressed before loading on the card. The resulting byte code density can be smaller than native, if applets are optimally coded programs.

4.2.2 Security

The security of the VM is first of all based on the public scrutiny of the specifications and its use in internet browsers. In the early phase of the development, a number of security problems were detected this way. Technically the security is based on language and VM features. The absence of pointer arithmetic ensures that no unauthorized access to objects is possible through pointer forging.

The standard VM can only guarantee the security if the JBC is guaranteed to be well constructed and well behaved. This is why the VM contains a class file verifier. The class file verifier verifies all class files before execution through a 4 phases process. Since the verification of the class file is a complex task, it was not included in the Java Card. Instead all of the tests that do not require run time information are performed outside of the card and the resulting byte code is signed. The loader in the Java Card only loads class files which have been properly signed and the VM performs the run time checks when the application is started.

In addition, since the standard classes that are available through the API are implemented in native code, the security mechanisms of the VM are bypassed when executing these classes. As a consequence, the overall security of the card also depends on the security of the implementation of these standard classes. It is unclear how this split affects the overall security of the platform.

Certification services for the VM and the standard classes are available from SUN. All licensees must pass all mandatory VM and API tests in order to use the Java brand logo.

4.2.3 Target requirements

The Java language and the JVM instruction set were developed with a 32 bit machine and a large memory space in mind. The derived specifications for Java Card took a number of resource considerations into account, but the resulting VM is still 32 bit oriented. Due to the object-oriented nature of the VM and the run time checks, a high performance smart card chip is needed to obtain acceptable performance.

Today's Java Card platforms are based on 8 and 32 bit CPU's. It should however be noted that most vendors agree that more than an 8 bit platform is needed. IBM however states

that an 8 bit processor is sufficient. A MMU is not needed due to the fact that the application data is separated by object access permissions and not by application firewalls.

4.2.4 Memory requirements

Due to the object-oriented nature of the VM, the representation of data and code requires additional space to store the object related information. This results in larger than necessary data and code space. However, also due to the object-oriented nature, code is in general better factorized which results in smaller code. The amount of overhead depends on the class file definition, the VM implementation and the application programmer. Currently no data are available. A hands-on design exercise is currently being performed to quantify the overhead.

The required code for the VM and the underlying OS is approximately 16K, while the standard Java Card API is about 8K.

4.2.5 Hardware implementations

A software implementation of the VM on an 8 bit CPU, yields about 3000 instructions per second. This results in slow applications. This is why a number of vendors are contemplating or designing a hardware implementation of the VM. IBM claims however to have implemented a VM on an 8 bit platform which delivers adequate performance (worst case 25% slower than native applications).

Another approach is the design of processors which are more suited to execute high level languages such as Java and C. A number of these designs are based on 32 bit RISC CPU's. No concrete data about the processors and the resulting speed gain are available at this moment.

4.3 Product Issues

4.3.1 Specification

The Java Card VM is based on an open specification, which is controlled by SUN Microsystems. The platform was developed under guidance of card manufacturers and standardized by SUN Microsystems. The evolution of the platform is mainly driven by an industry forum called the Java Card Forum.

Licensing is controlled by SUN Microsystems. A license is needed to implement the VM and consists, on a standard basis, of a lump sum (approximately 400.000\$) and a royalty per card (3 - 5 %). There are options for different pricing models; official pricing is given out only under NDA with SUN. Application licenses are not needed. Current Java Card licensees (more than 30) are Bull, Citicorp, Dallas Semiconductor, De La Rue, Gemplus, Giesecke & Devrient, IBM, Inside Technologies, KeyCorp, Motorola, NatWest, ORGA, Schlumberger, and Visa.

Telecommunication companies are driving the specifications to a much larger extent than the financial sector. This results in more products targeted at the telecommunications market than at the financial market.

4.3.2 Platform

The platform is multi-sourced. Table 3 gives an overview of the Java Card platforms available from different vendors.

4.3.3 Development Tools

The development tools for standard Java can be used to develop applications for Java Card. The resulting byte code can be simulated at two levels: before and after compressing the class file.

- **Standard byte code:** Before compressing the class file any standard development environment can be used. This means that a large number of tools including compilers, debuggers, assemblers and simulators can be used. Only an implementation on the simulator of the Java Card API 2.1 class is needed in addition to the classes already available in the standard development tools.
- **Compressed byte code:** Each vendor built a platform specific post-processor based on a public reference converter, that translates the standard Java class file into a more compressed format (the cap file). Since the Java Card Forum dd. 3-4 December 98, this format has been standardized for the 2.1 release (24/2/99), but is not yet implemented in many products. The only product that adheres closely to the specification is the Gemplus GemXpresso RADII 211. (The G&D [Sm@rtCafé](#) toolkit 1.1 is not yet fully compliant to this specification).

4.3.4 Certification

For security reasons and due to the features of the VM, only the certification of the VM is needed. Applications do not need to be certified since the VM guarantees the necessary security features. Certification services for the VM and the standard classes are available from SUN. All licensees must pass all mandatory VM and API tests in order to use the Java brand logo.

4.3.5 Usage

At this moment, Java Card is coming out of the development and research phase, and is used mainly in GSM applications. By March 2000, more than 10 million Java Card have already been shipped.

4.4 System Issues

Some system houses are beginning to offer back office and terminal infrastructure solutions for loading applications in the field. All card vendors are concentrating on the card platform.

Visa is addressing the card initialisation and personalisation, secure downloading and life cycle management issues in the Open Platform Specification. Currently Visa is working together with the Java Card Forum to incorporate the specifications in the official SUN Microsystems specifications. Several companies provide backend card management systems: ActivCard, Platform 7, Cards Etc, Oberthur, ...

4.4.1 Java Card Forum

The Java Card Forum is an industry forum to discuss issues and to drive the evolution of Java Card and the required API's. Proposals for foundation of the forum were initiated by Gemplus and Schlumberger in Feb 97 and the forum was established in April 97.

Current Java Card Forum members are Bull, Citibank, De La Rue, Gemplus, Giesecke & Devrient, Hitachi, IBM, Inside Technologies, KeyCorp, Motorola, NatWest, NEC, Oberthur, ORGA, Schlumberger, Sermepa, and Toshiba. Technical committee observers are Dallas Semiconductor, IPM, Lucent, Security Dynamics and Sun Microsystems.

Currently Europay has an observer status along with Amex, MasterCard and Visa.

The forum has developed a subset of the Java language and byte code for Java Card and a standard API. The proposals were adopted and turned into a Java standard by SUN Microsystems. At this point additional API's are being defined for different vertical market segments.

4.4.2 Application Loading

The Java Card v2.1 specification is very open on this topic. The only hard requirement is that the on-card installer be selectable. Combining that with the fact that only 2 native classes are defined in the specifications, one comes to the conclusion that there are 2 ways an application could be loaded on the card:

1. The Java Card VM does not expose any install or memory management functions. This is a situation very similar to the one for MULTOS. However, unlike MULTOS, there is no security infrastructure defined, so the security of the application load is very much at the discretion of the platform vendor. Indeed, OP 2.0 defines only the principles. The vendor must determine by himself the implementation details that his system requires.
2. The Java Card VM does expose the install and memory management functions in a native API. In this case, the issuer can control 100% how applications are loaded, but the downside is that he needs to verify that none of the applications he allows on the card access those classes. Indeed, a valid application could contain a Trojan horse that could be used to load applications without the user knowing about them. The Java Card Forum and SUN are currently in the process of standardized memory and applet management models.

4.5 Summary - Requirements Matching

4.5.1 Dynamic application management

Dynamic application management is possible with Java Card 2.1. However, there is no definition of secure card management (loading / execution / locking / unlocking / deleting of card applications / software) in Java Card 2.1, and hence the use of auxiliary, proprietary specifications is needed. One of these specifications is e.g. Open Platform 2.0, which leaves a large amount of flexibility and responsibility to the issuer and the card vendor to define, implement and approve the security.

4.5.2 Issuer involvement

The issuer can rely on the platform to keep applications safe from each other. Depending on the way the application management is implemented, the issuer may need to scrutinize third party applications that are loaded on his cards.

4.5.3 Security

At this point in time, no security approval has been granted to any Java Card.

4.5.4 Model

Java Card 2.1 can be used in both issuer centric and cardholder centric models, because there is no fixed, mandated security mechanism for application management.

4.5.5 Cost

Java Card 2.1 implementors become scarce, as it is a large investment to implement the VM. The general model seems to be that there are one or two parties that make the VM and license it to several card embedders.

As an indication, we expect the following price trends to occur for a 25K EEPROM Java Card 2.1 in minimal batches of 20.000 cards, with a total of 1000000:

Year	Without Crypto	With Crypto
2000	6.14 EUR	8.01 EUR
2001	5.22 EUR	7.37 EUR
2002	4.70 EUR	6.27 EUR

In general, the loading security and application personalisation cost are dependent on the time spent on the manufacturer's equipment (be it for the actual card interaction or for the generation of the card load certificates), and adheres the following rules:

- ❑ At Card Manufacturing, during the initialization process, this is then a fixed cost (approximately 3000 EURO per batch).
- ❑ At Personalisation Center, this is a variable cost and can be done in batch mode
- ❑ After issuance: the largest cost component will be the applet server and card management infrastructure.

4.5.6 Open vs. proprietary governance

Java Card is wholly controlled by SUN Microsystems. The specification can be influenced through the Java Card Forum, in which the banking industry is well represented.

4.5.7 Cross industry support

Java Card has taken off much stronger in the GSM industry. Java Card has been adopted by ETSI and incorporated into specifications from the ETSI SMG9 committee, namely ETSI TS 03.19 (SIM API for Java Card), and TS 03.48 (Over the air loading of applications, based on the Open Platform API)

4.5.8 Availability

So far, only one 100% version 2.1 card is available, since Cartes'99 (GemXpresso RADII 211). This toolkit is also nearly fully compliant with OP/VOP2.0.

5. Windows for Smart Cards

5.1 Overview

In the summer of 1998, Microsoft has launched a new business unit promoting a chip card OS, named 'Windows for Smart Cards' (WFSC). The version 1.0 is available since Q1 '00.

The platform model is mixed:

- The terminal can call applications written in OS-dependent language. The security is achieved by the fact that only Microsoft is able to write and download such applications onto the card. Currently, only basic ISO 7816 commands seem to be available. This system's functionality is comparable to the traditional systems of the different vendors and as such yielding equivalent memory / cost as the current existing operating systems as e.g. available from Gemplus, ORGA, Schlumberger, De La Rue, Oberthur...
- The terminal has direct access to files stored in the card. In this case, the card acts as a file server.
- The terminal can call applications written by third parties. As they aren't verified by Microsoft, they are interpreted by a virtual machine called 'Run Time Environment' (RTE). Microsoft started from scratch to create the APIs (independent from Windows CE).

WFSC claims to be a multi-application platform. However, due to the fact that

- WFSC is characterised by a lack of context management information (i.e. current selected application) and due to the fact that
- if two applications on the card can accept the same command, the processing must be the same for said command for both applications (e.g. loaded applications may influence the way of programming applications to be loaded),

WFSC isn't a fully multi-application platform in the sense we defined it in chapter 3.

The development system consists of an environment to both create a custom version of the operating system and create applications. The operating system is put into ROM while the applications can be put into ROM or EEPROM.

The target applications include:

- Identification, authentication, encryption, decryption: Windows 2000 logon, e-banking, e-mail, e-commerce
- Closed user groups: campus schemes, loyalty, citizen cards
- GSM

- Finance: debit/credit, E-purse
- Pay-TV
- Transportation (contactless)

5.2 Technology Issues

5.2.1 Virtual Machine

The current system consists of a library of operating system functions that can be assembled in a modular fashion. The user selects the required functions and security level in his development environment and compiles the operating system functions into an executable which can be put in ROM. This can be done for multiple target platforms with different target chips.

The virtual machine is optional in this system.

5.2.2 Security

The security of the applications depends on the type of platform model used.

The security of applications written in OS-dependent language is achieved by the fact that only Microsoft is able to write and download such applications onto the card. In such a model no virtual machine is used, therefore the security of the applications depends on the inspection of the code by Microsoft.

The security of files is guaranteed by access control lists (ACLs).

The security of the applications unverified by Microsoft is guaranteed by the virtual machine.

5.2.3 Target requirements

The target IC should have an 8 bit CPU with 32K ROM, 32K EEPROM and 1K RAM. Based on the user requirements for cryptography a cryptographic coprocessor is required if RSA is included. A MMU could be needed to partition the memory into different independent sections depending on the user requirements and the way the user wants the applications to be certified.

5.2.4 Memory requirements

Europay is currently investigating this with the card manufacturers in order to get this information.

5.2.5 Hardware implementations

Hardware implementations are not considered for the moment.

5.3 Product Issues

5.3.1 Specification

Microsoft manages the specifications of the platform. They are intentionally not public. For the moment, the specifications aren't fully specified yet.

5.3.2 Platform

It is the intention to multi-source the OS to the different existing smart card vendors.

Card manufacturers pay Microsoft a licensee fee per card running WFSC. Microsoft discusses only pricing with its licensees. Card manufacturers determine independent of Microsoft the price of the smart card. Issuers can buy WFSC directly from a card manufacturer.

As Microsoft want the operating system to be used in the mass volume low price markets like for example debit/credit cards, it is clear that the price of the OS should be according. In line with Microsoft other platforms, the cost of the operating system is only a small fraction of the price of the computer system. Microsoft has also made a distinction in pricing between the version of the OS supporting RSA and the version not supporting RSA (lower price), which will be used more in the mass volume markets.

The first OS prototype was available on SLE66CR from Infineon. Microsoft has today 2 licensees: Gemplus (contact card with ATMEL 32 component) and Sagem (contact card with flash memory). Both are in sales mode. Microsoft will make implementations on the more common processors: the Motorola 6805, the other (smaller) Infineon devices and the Hitachi H05. Later it will also become available for the ST range of products.

5.3.3 Development Tools

As application development tools the existing Visual Basic software development kit will be used. In addition, there will be support for a C API (only on the terminal side) and support for PC/SC.

5.3.4 Certification

Given the modular approach of the OS, it is difficult to certify the OS since there will not be a single version of the OS but a whole range of customised OS'es. Microsoft is willing to assist (financially, expertise, resources) those customers that want to take it to certification pending negotiation.

5.3.5 Usage

The complete product is still in development phase. However, two implementations of the platform are today in sales mode.

5.4 System Issues

Microsoft does not address this issue. WFSC is only a platform; a business model to choose a complete back office for downloading applications isn't imposed nor advised. Consequently, WFSC can be used in a cardholder centric, issuer centric or PC centric model.

5.5 Summary - Requirements Matching

5.5.1 Dynamic application management

Dynamic application management is possible with WFSC. However, there is no definition of secure card management (loading / execution / locking / unlocking / deleting of card applications / software) in WFSC, and hence the use of auxiliary, proprietary specifications is needed.

5.5.2 Issuer involvement

The issuer needs to define his security mechanisms for the card management.

5.5.3 Security

It is up to the card manufacturer or issuer to determine whether to the platform is required to be ITSEC or CC certified.

Microsoft would like implementations to be ITSEC E4 certified but no certification has been obtained yet.

5.5.4 Model

As WFSC is only a platform, a business model isn't imposed nor advised. Consequently,

- WFSC can be used in a cardholder-centric model,
- WFSC can allow an issuer to be fully in control of the content of the card,
- WFSC can be used in a PC-centric model.

5.5.5 Cost

No information is available at this moment to make a valid statement.

5.5.6 Open vs. proprietary governance

The specification is 100% proprietary to Microsoft.

5.5.7 Cross industry support

Microsoft plan to enter every smart cards markets: GSM, finance, identification authentication, encryption, decryption, loyalty, transportation, ...

5.5.8 Availability

The availability of the complete version 1.0 specification is imminent. Only a part of the version 1.0 specification is available.

6. MULTOS

6.1 Overview

MULTOS is a secure operating system which has been specifically designed for smart card systems. MULTOS is responsible for managing the interaction of application, the MULTOS Security manager, the basic processor functions and the cardholder with each other. Each entity can send events to MULTOS which it will translate in a series of events for other entities or handle the event itself when the requested (system) resources are under its own control.

6.2 Technology Issues

6.2.1 Virtual Machine

The virtual machine in use with the MULTOS operating systems is currently a MEL VM. The MEL instruction set has 31 instructions and 2 addressing modes. These cover system control, flow control, and stack, literal, unary block and binary block operators. In addition there are mandatory and a limited number of optional primitives. The instruction set together with the mandatory primitives is sufficiently rich to code applications efficiently.

The address space is divided into separate code and data spaces, and are both limited to 64K each. MULTOS guarantees that private data is visible only to the owner of the data. MULTOS provides the possibility to pass data from one application to another application through the use of a public buffer and a mechanism called Delegation. Application separation is therefore good.

6.2.2 Security

The security target for the MULTOS Platform is ITSEC E6 high (see attached certificate), and this has been achieved. The OS manages all system resources and controls the access. The VM relies on the OS to achieve waterproofing between applications.

6.2.3 Target requirements

Current implementations use 8 bit CPU's. Questions were raised that those cannot be considered to be sufficiently powerful to be used for an interpreter. Measurements however show that the current family of Hitachi and Infineon (8 bit) have sufficient power for the applications in the field today.

A cryptographic engine (RSA co-processor) is mandatory since the MULTOS load and delete mechanism uses asymmetric cryptography. An MMU is not mandated.

6.2.4 Memory requirements

The VM requires 4K, the executive 7K and the libraries (including EMV functionality) 7K.

6.2.5 Hardware implementations

Hardware implementations are currently not planned due to the stringent requirements for ITSEC E6 HIGH evaluation. The ITSEC criteria for software design are well defined, and the criteria for hardware evaluations has been specified. If the system is implemented to a large extent in hardware, it may prove difficult to prove a number of basic features to ITSEC specifications and need to be assumed. This could weaken the confidence in the overall system.

6.3 Product Issues

6.3.1 Specification

The MULTOS specifications are open and owned by Mondex International. MAOSCO is the industry wide consortium that has been set up to control the evolution of the specifications. Licenses are required from MAOSCO to implement the MULTOS operating system, including the underlying OS, VM and load/delete mechanism. No further royalties are needed per card.

6.3.2 Platform

The platform is multi-sourced. Today there are however only three different OS providers implementing a version of MULTOS. Other vendors license this platform. Table 3 gives an overview of the different platforms available. It should be noted however that all currently available systems use the Dai Nippon Printing or KeyCorp implementation. In addition, the following vendors will offer MULTOS cards and applications: AMMI, Bull, G&D, Gemplus, GPT, ICT, Motorola, Orga, Schlumberger, SCS, Security Plastics and US3.

6.3.3 Development Tools

Currently there are three sources for the development tools:

- **GTI** has the MULTOS Development Suite available which includes an assembler, a simulator and a debugger. In addition a very low performance C compiler is available. Under development are a card loading and application personalisation module, a terminal simulator, a card probe terminal interface and an improved C compiler by Q2 2000.
- **Hitachi** has a MULTOS Application Development Tools which consists of the MULTOS Assembler and Linker (MALT) and the MULTOS Simulator and Debugger (MSDT). These tools have been announced in September 98, but have not been evaluated thoroughly because of their incompatibility with the others. A C compiler will be launched later this year.
- **SwiftCard Technology** has developed a toolkit that includes assembler, debugger and optimised C compiler. A beta version of a Java compiler are also available and product is due to be launched in Q2 2000.

6.3.4 Certification

Waterproofing between the applications is guaranteed by the kernel. This means that the applications do not need to be certified.

Given the time consuming and detailed security evaluations required by the ITSEC process, there will only be limited number of certified platforms available to the market.

6.3.5 Usage

Currently the system is only used in several rollout projects and small pilots, with an estimated total of 3 million cards in the field by the end of 1999. From the known projects and the likelihood that they will proceed, one can venture that around 22 million MULTOS cards will be issued by end of 2000.

6.3.6 System Issues

The MULTOS Certification Authority (CA) addresses the system issues such as application loading & deleting, load/delete certificate generation, OS key management etc.

An application Load /Delete Certificate needs to be purchased from the MULTOS CA for each application and each card (0.025 GBP). Application code and associated data can be encrypted before transmission to the card over a possible insecure network. The card decrypts the application after it has been received. The card and the application mutually authenticate each other before the application is accepted.

6.3.7 MAOSCO

MAOSCO is a consortium formed in May 1997 to drive the adoption of MULTOS as an open industry-controlled standard. The 12 members of the consortium include DNP, Europay, Infineon (formerly Siemens), KeyCorp, Motorola, Hitachi, Mastercard and Mondex sharing one seat, the Fujitsu/ICL/Amdahl group, G&D, Novus Discover, Amex and Telstra. The control of the MULTOS specification is contractually vested with the consortium members preventing any organisation from manipulating the specification for their own commercial benefit.

It is expected that the consortium will work on libraries for GSM, Elliptic Curve Cryptography (ECC) and contactless interfaces plus a number of small additional features, which will eventually result in a version 5 specification mid 2000.

6.4 Summary - Requirements Matching

6.4.1 Dynamic application management

This is definitely possible from the card point of view.

6.4.2 Issuer involvement

The issuer can manage applications on his card

6.4.3 Security

The existing implementations have either achieved ITSEC E6, or are in the process of doing so.

6.4.4 Model

The model is exclusively issuer centric.

6.4.5 Cost

We have been informed that for equivalent hardware capabilities, the additional cost for MULTOS would be insignificant. There is a fee of 0.025 GBP per application load or delete certificate.

6.4.6 Open vs. proprietary governance

The specification is 100% controlled by MAOSCO, a cross-industry consortium. MULTOS is licensed on an open and non-discriminatory basis.

6.4.7 Cross industry support

Currently MULTOS is only used in the banking environment, but efforts are ongoing to put it in use in the GSM and UMTS markets.

6.4.8 Availability

Test cards are available for application developers and live cards are rolling out.

7. Matching The Requirements

Requirement	MULTOS	Java Card	Windows
Dynamic application management	Possible	Possible	Possible in some implementations.
Issuer involvement	Not necessary	Vendor defined	Vendor defined
Security	ITSEC E6	Private requirements	Microsoft approval
Model	Issuer Centric	Free	Free
Cost*	FC + LC	FC + SC + RF	RF + ?
Open vs proprietary governance	Open in consortium	Proprietary (SUN)	Proprietary (MS)
Cross industry support	Finance and PKI	GSM	PC
Availability	Production (V3 & V4)	Production (V2.0, samples V2.1)	Samples

*: roughly speaking, the fixed cost (FC) for either MULTOS or Java Card are similar and largely depending on platform and volume rather than OS. The Load Certificate Cost (LC) for MULTOS is well defined, the Security Infrastructure Cost (SC) for Java Card is an unknown quantity and will probably be vendor dependent. A Royalty Fee (RF) is payable for both Java Card and Windows. MULTOS is royalty free.

8. Benchmark

This section presents a summary of the results of the development of an EPI ‘Debit Credit Off the Shelf’ (spec version 3.03) card application on the Native C, MULTOS, Java Card and Windows for Smart Card platforms. The goal of this exercise was to compare the Native C, MULTOS, Java Card and Windows for Smart Card platforms using a real-life application, which guarantees the same behavior, plus the platforms are based on the same hardware component, i.e. the Infineon SLE 66CX160S.

8.1 Development environments and components used

8.1.1 Native platform

The application was written in C with some low-level routines in Assembler using the Keil μ Vision/51 version 1.13 development environment. The application was tested with SLE 44C80S boundout and SLE66CX160S soft mask card.

8.1.2 MULTOS

The application was written in MEL and C:

- In MEL, using the MULTOS v3 and v4 specification using the MULTOS Development System (MDS v1.24).
- In C, using SwiftCard development environment.

We also had a look at the Hitachi MULTOS v4.0 development tools for H8/3114. The application was tested on the Hitachi v3 and v4 cards (based on the H8 component) and the Keycorp v4 cards (based on a Siemens processor, the SLE66CX160S).

8.1.3 Java Card

The application was written in Java against the Java Card 2.1 specification using the combination of the Microsoft J++ and Giesecke&Devrient Sm@rtCafé (1.1) development environment. The application was tested on the Giesecke&Devrient cards included in the development kit. These cards are based on the Siemens SLE66CX160S component.

Europay is currently developing the same application using GemXpresso RADII 211 jointly with Symantec VisualCafé v3.0.c. GemXpresso cards are based on Phillips P8 component.

8.1.4 Windows for Smart Card

The application was developed in Visual Basic using the WFSC APIs v1.0. The development kit consists of the traditional Visual Basic 6.0 development's environment with plugs-in added in order to get smart cards tools.

8.2 Native development

8.2.1 Keil environment

Development in C

- The major part of the application development was coded in C language
- The definition and location of variables are specific to this environment.

The major problem encountered during the development with the Keil Simulator platform was the fact that the real time behaviour is not accurately simulated on a PC, so exchanges with a card reader do not work correctly. Only the Emulator (Kscv) or bondout allow to perform the real time tests.

Development in Assembler

- The DES algorithm and some low level routines (BCD addition, string comparison, ...) were developed in assembler to improve execution times and code efficiency , and to avoid using the standard libraries which were not optimized for our application.

Debugging

- The C debugger allows viewing C and Assembler code generated.
- A Performance analyser provides the timings of functions
- No tool to build commands (or scenarios) exchanged between the card and the external world.

Loading on bondout

The load of application in the Siemens bondout is performed in two steps

- Generation of an EEPROM which contains the code of the mask including the application, the dispatcher, the cryptographic routine the Siemens routines for I/O and EEPROM read /write
- Personalisation of all data included in the file system in the ICC EEPROM using proprietary commands (scenario with Update Binary commands)

Loading the softmask

To load the DC application in the Siemens soft mask card several actions have been done:

- Replace all MOVC assembler instruction by the MOVX to access to the EEPROM and not to the ROM
- Include the personalisation data in the code segment to have an initialisation at compilation

- Set the location address in linker for the soft mask
- Delete the first line in hex file (which contains at Address 0 a jump to the first code instruction)
- Load the code (hex file) using the Infinion tool in soft mask (cct op for SmartMask). This mask may be permanent or active for a limited number of reset; in all cases we include a command which recovers the initial OS of the card.

8.3 MULTOS development

8.3.1 Development in MEL

- Instructions on byte, word or multiple bytes allow the development of compact code
- MULTOS offers no support to work with BCD numbers
- All data transferred on I/O shall be buffered in RAM buffer
- To improve the performances some temporary data are stored in the public buffer (buffer for the input/output buffer), because the space in RAM for session data is small.
- All cryptographic libraries (DES, 3DES, RSA) are provided as standard.

8.3.2 MDS environment

C compiler

The MDS environment in principle comes with a C compiler. However the size of the generated code is 4 times that of hand written MEL code so that in practice this C compiler is useless. There is also no symbolic C debugger.

Development

- the development environment runs only under Window 95 but not under NT
- A single C source for a project (which can include several other ones) which generates an Assembly file
- A single Assembly source file for a project (which can include several other ones) => an update in a source file requires a compilation of all files
- Assembly errors are reported in a log file and not directly in source file (and with an error on the line number)
- Compilation is slow (1 minute for the compilation of our application)

Debugging

- The debugger is not able to simulate the real size of RAM and EEPROM of MULTOS Card ; so some problems were discovered during the execution on real cards (especially all problems with stack which make the card mute)
- The breakpoints are not easy to set because they shall be set one the single lst file of the project.
- The debugger is slow (~1 minute for the execution of the Generate AC).

- It is not possible to test the cryptographic functions of the application using the simulator because the primitives DES, MAC are not implemented (they return the input value or a null value).
- The simulator has bugs in some primitives (MAC primitive corrupts the stack).
- No performance analyzer (it is not possible to estimate the timing of a command).
- The terminal simulator does not allow to build easily a scenario containing several commands.

8.3.3 Hitachi development environment

We only had a quick look to this environment but we have nevertheless found the following.

- There are some incompatibilities with the MDS environment on source file level
 - Data declaration
 - Procedure declaration
 - Mnemonics of some instructions (conditional branch, return, ...)
- Additional functions of the Hitachi debugger
 - Breakpoint in source file
 - Debug command STEP OUT
- Additional functions of terminal simulator of the Hitachi development environment
 - Scenarios can be used and a tool exist to build these scenarios
 - Two kinds of terminal simulator:
 - integrated in the development environment to simulate the input of APDU
 - to be used with a real smart card
- There is no integrated editor.

8.3.4 SwiftCard development environment

The SwiftCard development environment allows developing applications for MULTOS cards both in C, Java and MEL. A Beta test version was used to develop our application, but the first release of this product is now available (v1.1).

Development

- The development environment does not provide any editor and the compiler runs directly under DOS
- Errors are reported in the current DOS window and not directly in source file.
- // is not supported as a comment line
- The compiler has an option to optimize the size of the executable file. It permits to gain around 1%.

Debugging

- The C debugger only allows viewing the segment data but it does not allow reading and updating variables.
- The debugger is not able to simulate the real size of RAM and EEPROM of MULTOS Card; so some problems were discovered during the execution on real cards (especially all problems with stack which make the card mute).
- No performance analyzer (it is not possible to estimate the timing of a command).
- The terminal simulator does not allow building easily a scenario containing several commands.

8.3.5 MULTOS Application Loader

The MULTOS Application Loader provided by MAOSCO has been used to load applets in MULTOS cards.

To load and unload applets in MULTOS cards, certificates provided by MAOSCO are mandatory (even for test cards).

The certificate to load an applet in test cards contains the size maximum of code, RAM session data and EEPROM personalisation data. When the code size or the personalisation data exceeds the size defined in the load certificate, the load of the applet aborts; but when the size of session data exceeds the size defined in the certificate nothing is detected during the load. However, during the execution of the applet, when a session data variable outside the area reserved by the certificate is read or written, the card becomes mute.

Also, it was difficult to define the right parameters of the certificate for loading the Debit / Credit application on MULTOS V3 cards because the code, the RAM and the EEPROM data have to be optimized to be loaded (and these optimizations shall only be tested in the real card because the MDS environment does not simulate the real size of the stack).

8.4 Java Card development

8.4.1 Development in Java

- It is important to define variable as transient to store them in RAM (if not they are stored in EEPROM)
- Java Card offers no support to work with BCD numbers
- Currently the application code is not really object-oriented because the original application developed in native C code for Siemens has been ported rather straightforward to Java Card. Since the same approach has been followed for MULTOS, this has the benefit of making the MULTOS and Java Card applications very easy to compare.

8.4.2 G&D Sm@rtCafé environment

Development

- The crypto interface provided by Sm@rtCafé is NOT compliant with Java Card 2.1
- The application code is written in Java and compiled into a Java class file using Microsoft J++. This Java class file is then further compiled into a Java Card cap file using Sm@rtCafé.

Debugging

- A simulator is provided which behaves exactly as the physical cards, supports scenarios and is easy to use.
- The byte code is not displayed properly (only the beginning of the file is displayed).
- The simulator aborts sometimes when the operator makes a mistake.
- The sources edited with MS J++ contain abnormal end-of-line characters which causes the Sm@rtCafé source level debugger to hang. This problem disappears if the offending end-of-line characters have been replaced by ordinary ones by another editor.
- A performance analyzer is missing so it is difficult to know the performance of the function programmed for a card application.

No tool to obtain the size of each function.

Loading

- The G&D Sm@rtCafé development environment for Java Card contains the loading applet function. All scenarios prepared with the PC simulator can be reused with the real card.

8.4.3 GemXpresso RADII 211 environment

The execution of this software combined with Symantec VisualCafé is extremely slow. It requires a minimum of 128M RAM.

Development

- Europay is currently developing the Off-The-Shelf applet for Java Card using this environment.

Debugging

- The debugger's analysis is outstanding.

Loading

A tool to evaluate the applet size before loading it onto a card is available. The GemXpresso RADII 211 environment for Java Card contains an application loader.

8.5 Windows for Smart Cards development

8.5.1 Development in Visual Basic

- Variables that must persist after power off are stored in files. Otherwise, variables that must be stored in RAM are declared in the code.
- The APIs doesn't provide easy operations on arrays.
- The bytes read from a file are put in the array indicated in argument, by starting at the offset 0. If these bytes had to be copied to an offset different from 0, it must be done one by one into the correct offset later on. Time is wasted (two copies in place of one) and space is wasted because of the need of an 'intermediary' array.
- The shortcoming raised in this section is similar to the previous one. In order to write bytes to a file, these bytes should be at the offset 0 of an array. If the bytes you want to store in a file are at an offset different from 0 of an array, you must first of all copy these bytes at the offset 0 of an array.
- Remark: ISO7816 commands written in OS-dependent language by Microsoft were not used, as they are too generic and not tunable to our use.

8.5.2 Microsoft Visual Basic 6.0 environment

Development

- The application consists of a single Visual Basic project.
- All APDU commands that can be accepted by the application are re-directed by the system to this application. By the way, no other application which could accept these APDU commands can be loaded on the card.
- The application is not validated yet because of unavailability of Triple DES on development cards (v1.0) and because of loading problems (see Loading section).

Debugging

- The debugger's analysis is outstanding.
- The OS-dependent code produced from the Visual Basic code's compilation is not known. There is no tool to get the size of each function.

Loading

- Europay is not able to download an application onto a card yet. The origin of the problem seems to be an incompatibility with Windows NT (v 4, service pack 5) used at Europay, so that reader's drivers do not install correctly.

8.6 Comparisons between environments

8.6.1 Comparison between Sm@rtCafé and MULTOS MDS environments

- The G&D Sm@rtCafé development environment for Java Card is more user friendly than the MDS development environment for MULTOS, but the difference is less pronounced with the Hitachi environment for MULTOS (to be confirmed because the tests with Hitachi have been done very quickly).
- Developing an application using Java is roughly 2 times faster than using the MEL-MDS environment for MULTOS; this is largely due to the language but also to the development environment.
- The simulation fully reflects card behavior in Sm@rtCafé; but the crypto functions are not available in the MDS environment (has to be checked for the Hitachi environment).
- A Performance Analyser is missing in all development environments.

8.6.2 Comparison between Keil C and SwiftCard C development environments

- The Keil development environment includes more development tools than SwiftCard. (Performance analyzer, the capability to read and set variable)
- The SwiftCard development environment has an input message facility, the Keil environment has not.
- The SwiftCard development environment fully emulates the MULTOS primitives; the Keil environment only emulates a SLExx component.

8.6.3 Comparison between Sm@rtCafé and SwiftCard C development environments

The G&D Sm@rtCafé development environment for Java Card (with Microsoft J++ added) is more user friendly than the SwiftCard development environment for MULTOS:

- The Java editor and compiler are integrated in a window environment but not the SwiftCard one.
- The Sm@rtCafé debugger allows to view and update all variables but not the SwiftCard one.
- The Sm@rtCafé tool allows to build scenario with multiple commands, SwiftCard tool only includes a tool with a single input command.

But a Performance Analyser is missing in these two development environments.

8.6.4 Comparison between Sm@rtCafé and GemXpresso RADII 211 development environments

Europay plan to compare these Java Card environments.

8.7 Implementation Data

8.7.1 Code Size

The code size for the Debit/Credit application:

- 5.6 Kbytes for the MULTOS C
- 5.5 Kbytes for the Native C platform (without CMS, DES and I/O routines)
- 5.7 Kbytes for Java Card
- 4.2 Kbytes for MULTOS MEL
- 29 Kbytes for WFSC.

8.7.2 Application Validation

The Off-The-Shelf application running on Native C (using the bondout and the soft mask), the MULTOS v3 and v4 cards and the Sm@rtCafé have been successfully tested using the Europay tool for type approval.

This tool verifies all the application behaviors (card risk management, cryptography, script command).

The application implemented on WFSC has not been approved yet because of application size's wideness.

8.7.3 Transaction Timings

The following table provides the card processing time for several kinds of debit/credit transactions. These timings have been made on a PC emulating an EMV terminal.

The timings include:

- the time to transfer the commands from the terminal to the card reader
- the time to transfer from the card reader to the ICC
- the processing time in the ICC
- the time to transfer from the ICC to the card reader
- the time to transfer the commands from the card reader to the terminal

The timings exclude:

- the processing time of the terminal itself.

The following table lists the total transaction times of four different transactions. The differences between the different transactions are:

- *Crypto enabled - crypto disabled*. When the crypto is disabled, no application cryptogram is generated. This allows us to measure the time needed for the crypto functions.

- *Cumulative amount test executed - cumulative amount test not executed.* The cumulative amount test is a Card Risk Management test without cryptography. This allows us to compare the speed of pure token execution.
- *On-line - off-line.* The on-line transaction includes all EMV functionality (including issuer authentication). It gives an idea of the maximum processing time.

Platform: Transaction description	Native C (Softmask) (SLE66CX 160S)	Native C (Bondout) (SLE44CR 80S)	MULTOS V4 (MEL) (SLE66CX1 60S)	MULTOS v3 (MEL)	MULTOS v4 C SLE66CX 160S	MULTOS v4 C H8MA410 b	Sm@rtCafé (SLE66CX1 60S)
Off-line transaction accepted - Crypto enabled - Cumulative Amount test executed	3.74		6.06	6.5 s	6.75 s	6.1 s	8.6 s
Off-line transaction accepted - Crypto disabled - Cumulative Amount test executed	3.61		5.8	6.3 s	6.51 s	5.77 s	7.3 s
Off-line transaction accepted - Crypto enabled - Cumulative Amount test not executed	3.66	3.9 s	5.73	5.8 s	6.39 s	5.9 s	8.3s
On-line transaction accepted - Crypto enabled - Cumulative Amount test executed - Issuer authentication	4.2		6.72	7.4 s	7.6 s	7.0 s	11.2 s

8.7.4 Command timings

The next table shows the timing for each command. The accuracy of each measurement is +/- 50 ms.

Command	Platform: Native C Softmask SLE66CX160S	Native C bondout SLE44CR80S	MULTOS v4 MEL SLE66CX160S	MULTOS v3 MEL	MULTOS v4 C SLE66CX160S	Sm@rtCafé Java Card SLE66cx160S
Selection of an application that is present	150 ms	150 ms	300 ms	170 ms	320 ms	300 ms
Get Processing Options	140 ms	140 ms	310 ms	440 ms	330 ms	300 ms
Read record (size 57 bytes)	185 ms	175 ms	230 ms	300 ms	250 ms	170 ms
Read record (size 116 bytes)	350 ms	325 ms	390 ms	360 ms	390 ms	310 ms
Get Data (PTC)	50 ms	50 ms	70 ms	60 ms	130 ms	110 ms
Verify PIN	110 ms	120 ms	120 ms	260 ms	190 ms	220 ms
Generation of AAC required by terminal	360 ms	460 ms	610 ms	800 ms	750 ms	1900 ms
Generation of ARQC required by terminal	380 ms	480 ms	920 ms	1200 ms	1200 ms	3000 ms
Generation of TC - cumulative amount check executed	400ms	500ms	1290ms	1430ms	1650ms	3210 ms
Generation of TC - cumulative amount check not executed	380 ms	500 ms	1020 ms	1170 ms	1170 ms	2860 ms
Generation of TC at 2 nd generate AC without Issuer authentication	400 ms	480 ms	770 ms	910 ms	1070 ms	2520ms
Generation of AAC at 2 nd generate AC with Issuer authentication failed	360 ms	470 ms	580 ms	760 ms	940 ms	2430 ms

The following table is a summary of the different timings and gives a comparison between the execution time of pure crypto functions and pure token execution on the Native C, **MULTOS** and Java Card platform.

Command	Platform: Native C Softmask SLE66CX160S	Native C bondout SLE44CR80S	MULTOS v4 MEL SLE66CX160S	MULTOS v3 MEL	MULTOS v4 C SLE66CX160S	Sm@rtCafé SLE66CX160S
Generation of the application cryptogram (only crypto: key diversification and MAC)	130 ms	<200 ms	300 ms	200 -> 350 ms		900->1300 ms
Cumulative amount check (only token execution)	20 ms		270 ms	260 ms	480 ms	350ms

8.8 ISO and EMV compliance issues

8.8.1 MULTOS

- The select response of a blocked application is not compliant with EMV. No way has been found to circumvent this problem.

8.8.2 Java Card

- When a Java Card receives a select command that does not correspond to a known applet and no applet is active, the Java Card Run Time Environment responds with SW1SW2 6999. EMV however specifies that the card should respond 6A82. In the case of explicit EMV application selection this has no consequences. But in the case of implicit selection the terminal must abort the transaction if it receives such a wrong status word. A work around consists of adding a bogus PSE applet that will return 6A82 upon selection (code size is 200 bytes for the applet).

8.9 Conclusion

The following table resumes the results of the development of Debit Credit application using the Native C, the **MULTOS**, the Java Card and the Windows for Smart Card platforms.

Aspect	Platform	Native C	MULTOS SwiftCard	MULT OS MDS	Sm@rtCafé	Windows for Smart Cards
Development Environment		++	+	-	+++	+
Code size		++	- for C ++ for MEL	++	-	---
Timing performance		+++	++ for MEL - for C	++	--	?
Validation using Europay CTA Tool		+	+	+	+	-

The G&D Sm@rtCafé Java Card development environment is really better than the MDS **MULTOS** development environment (Hitachi **MULTOS** environment not sufficiently tested yet).

Application development is faster with the G&D Sm@rtCafé Java Card environment than with the MDS **MULTOS** environment. The Keil environment provides a Performance analyser, but not tool exists to build commands or scenario to debug the application.

The Java Card code size is equivalent to the **MULTOS** C code size, and is 25% larger than the **MULTOS** MEL code size. It must be noted that the SwiftCard compiler is a good product. The WFSC code size is 5 times larger than the other solutions.

The G&D Sm@rtCafé cards perform badly on cryptographic functions compared to the **MULTOS** v3 cards. For the Generate AC a comparison between **MULTOS** and Sm@rtCafé (without crypto function) shows that **MULTOS** v3 is 2 times better than Sm@rtCafé (Some complementary optimisation should be studied to try to improve the process time of the Generate AC of Sm@rtCafé).

The **MULTOS** C has performance between 20 and 30% lower than in MEL for the Generate AC; but the same performance for the other commands like Verify, Get Data, Get Processing options and Read Record. For the Generate AC command, the C **MULTOS** card is two time Faster than the Sm@rtCafé one.

Also for the Generate AC, the Native C code (running on SLE66xx component) is 3 times faster than the **MULTOS** MEL one (running on **MULTOS** v4 SLE66xx cards).

We should be careful to compare the cryptographic performances of the platforms, because no test has been performed on Differential Power Analysis or similar attacks.

For other commands, the differences between **MULTOS** v3 and Sm@rtCafé are not significant.

For a complete transaction G&D Java Cards need on average something like 30% more time then Hitachi **MULTOS** cards. It is not clear whether this difference is due to the different components, the quality of the particular implementations of the respective platforms on these cards, or features inherent to the respective platforms.

Validation of the Native C on Siemens IC, Java Cards and **MULTOS** v3 and v4 cards proceeds correctly.

For **MULTOS** and Java platforms ISO/EMV compliance issues exist. For **MULTOS** there is no work around. For Java card there is, but at some cost.

Appendix A: MAOS Platform Overview

Table 2 gives an overview of the technology on which the different platform families are based while Table 3 gives an overview of the product features.

A.1 Technology overview

	MFC family	Java Card	MULTOS	Smart Card for Windows
Virtual Machine				
Technology	Executable	interpreter	interpreter	Executable / interpreter
byte code	N/A	Java byte code	MEL	MIL
Coexistence with native code or other VM		Yes, but memory eater and performance killer	Yes, but subject to ITSEC E6 evaluation	(Yes)
VM performance (instr/sec)	N/A	3000 2 to 3 times slower than native	3500 2 times slower than native	N/A
Security				
Level		Not defined	ITSEC E6	Not defined
How	Inspection	system	System	
Certification		Not defined	ITSEC	by customer
Target requirements				
CPU performance	300 k/s	>300 k/s	>300 k/s	300 k/s
word length	8 bit	8 - 16 - 32 bit	8 - 16 bit	8 bit

	MFC family	Java Card	MULTOS	Smart Card for Windows
memory technology (E2/Flash)		both	only E2 available now	Both
Crypto engine	No	(Yes)	Yes	No
MMU	No	No	No	No
Memory requirements				
VM	N/A	14K ROM	4K + 7K	6K
		8K without downloading and security features		
Libraries		8K ROM	15K (included crypto lib 11K)	

Table 2: Technology overview of the MAOS systems

() : to be verified

A.2 Product Overview

	MFC	Java Card	MULTOS	Smart Card for Windows
Specification				
Open/Closed	Open	Open	Open	Closed
Controlled by	IBM	SUN Microsystems	MAOSCO	Microsoft
Evolution	IBM	SUN Microsystems	Members of MAOSCO	Microsoft

	MFC	Java Card	MULTOS	Smart Card for Windows
Licensing				
Required	Yes	Yes	Yes	Yes
From	IBM	SUN Microsystems	MAOSCO	Microsoft
What for	Mask / linkable code	VM implementa- tion	Complete OS Specification including VM	Implementation
Who pays	Implementer	Implementer	Implementer	Implementer
Amount	As issuer requests	400.000 USD +70.000 USD yearly	250.000 GBP + 50.000 GBP yearly	?
Platform				
Multi sourced?	Yes	Yes	Yes	Yes
Cost		2.5 - 7 \$	4.5 - 7 \$	2-8 \$
Development Tools				
HLL compiler	Standard	Same as standard Java	C / (Java)	Visual Basic / (C++)
Assembler	Standard	Yes	Yes	Yes
Debugger	Standard	Same as standard Java +specific tools	Yes	Visual Studio development environment
Application libraries				
What		Java Card		
Functionality	File, Security, Secure Messaging	I/O, security EMV	EMV, codelets	I/O, file, security
Resource requirements		8K for EMV	7K	
Certification				
Certified	No	No	No	No
Standard	No	No	ITSEC	

Level	N/A		E6
By whom	N/A	SUN Microsystems	ITSEC CLEF

	MFC family	Java Card	MULTOS	Smart Card for Windows
Status				
Usage	Volume / Pilot	Development / Pilot / Roll-out	Development / Pilot / Roll-out	No
where?	D, NL, US	Europe, US, Asia	Europe, US, Asia	N/A
By whom?	GeldKarte, NL StudentCard, Chipper, AMEX TravelCard		Financial institutions	N/A
Applications		D/C, e-purse, loyalty, e-com, ID, GSM	D/C, e-purse, ID, loyalty, small data store	N/A
Which card volumes?		pilots: several thousands GSM: several millions	2-3 million end 1999	N/A
low cost version		sample level	No	N/A
Sample available for Europay		Yes	Yes	Yes
Roadmap of platforms				
OS features			Complete OS spec including support for EMV 3.1.1, dynamic load and delete of applications., issuer control of application loading, VM environment, common security model, common	Configurable by issuer. (chosen from I/O (T=0, T=1), crypto, FAT file system with ACLs, VOP, GSM, ISO libraries, Runtime envir.)

Libraries	initialisation Support for EMV 3.1.1., RSA, DES, SHA-1, modular arithmetic, atomic transactions, utility functions
-----------	---

	MFC family	Java Card	MULTOS	Smart Card for Windows
time scale			Support for contactless, GSM and diagnostics Support for Elliptic Curve Cryptography (summer 2000)	
target cost		4-7 USD		
Roadmap of underlying silicon				
How much memories		32K E2 Flash, 32K ROM, 2.3K RAM (if E2) or 1K RAM (if Flash)	V3 on H8/3112 V4 on SLE66CX160S V4 on H8/3114	24k ROM, 512 RAM, 2K E2
Crypto available		DES, 3DES, RSA, SHA-2, MD5	DES, 3DES, RSA, SHA-1, GSM A3A8 (June 2000)	Yes
time scale			32K E2 silicon (Q3 2000)	
target cost		7 USD		

Table 3: Product overview of the MAOS systems

Appendix B: Silicon Platforms

B.1 Summary

Suitable chips for interpreter based MAOS platforms are being developed by different foundries. These chips will become available over the next years and will be initially targeted at the high end market. Quality and cost considerations for large scale deployment demand the chip area to be smaller than 12-15 mm² which may require a further evolution of chip technology.

B.2 Platform requirements

B.2.1 Memory

MAOS platforms require a silicon platform that has a large amount of ROM, EEPROM and RAM resources. Typically around 16K is needed for the VM and the libraries. Additional RAM resources are needed for the VM above those for the application. The minimal platform requirements would be 16K ROM, 4K EEPROM and 256 bytes of RAM. This would however result in a multi-application platform with defeats the requirements for multi-applications since it would not be able to contain a lot of applications (no applications in ROM and 1-2 small applications in EEPROM). Practically however 24K ROM, 8K EEPROM and 512 bytes of RAM are a more realistic minimum to support the VM. This way application libraries and common applications can also reside in ROM, and a more realistic application in EEPROM. (The implementation of a Debit/Credit application in Java byte code has been reported to need 8K including data when an EMV library is available).

B.2.2 Processing power

Current generation IC's are not really powerful enough to provide fast interpretation of applications. Most vendors say that a 16 or 32 bit platform is needed. Some say an 8 bit platform will suffice.

B.3 Product Features

To be able to dynamically load applications in an open environment, the applications need to be signed. This means that asymmetric encryption is needed. Given the fact that RSA cannot be executed fast enough on a normal CPU, an RSA coprocessor is required.

To be able to dynamically load applications in a closed environment, private key encryption will suffice. This does not require a cryptographic coprocessor.

Table 4 gives an overview of the road maps of the IC's of the different vendors that have at least 15K of ROM, 4k EEPROM, 512 bytes of RAM and with or without cryptographic processor. It should be noted that this configuration results at present in large die sizes.

B.4 Future Trends

B.4.1 Memory

Most manufacturers acknowledge that the MAOS systems really require more memory than available on the market today. This is the reason why future platforms will offer more memory in all categories ROM, EEPROM and RAM. Some manufacturers are forced to use different non volatile memory technology due to shrinking problems (Flash instead of EEPROM). Another trend is the research on Ferro electric memory (FeRAM) which leads to more dense memory. This technology will remove the need for having different memory technologies on the same chip as it allows the designer to partition the memory into different functional memory types by simple configuration while providing at the same time one single global memory block to the CPU.

B.4.2 Processing power

For the CPU's two major trends can be identified:

1. **32 bit RISC platforms:** A number of vendors (Motorola, Hitachi, NEC, SGS-Thomson) and research projects (MASSC sponsored by the EC/ESPRIT) are developing 32 bit RISC platforms. A RISC CPU is a good strategy for platforms with large amounts of memory, and good compilers since it provides good flexibility towards all high level languages. A RISC approach reduces the complexity of the processor, but requires a larger amount of memory since one complex instruction is replaced by several less complex ones. In large systems this enables faster execution since the instructions can be more effectively pipelined. Given the slow clock frequency of smart cards, there is no real need to switch to RISC. Only in the case that an interpreter is used, the penalty of increased memory use is not so bad since only the code for the VM itself is in executable form. The RISC approach is not the optimal approach for smart cards where memory space is an issue, unless a VM is used.
2. **Hardware VM's:** A number of vendors are designing IC's that are specifically tailored to particular VM implementations (NEC, SGS-Thomson, Siemens). The development efforts have however been slowed down with respect to the beginning of 1998.

B.4.3 Technology

Larger memory and more powerful processors lead to very large die sizes. This means that there will be important fall-out problems unless the die size is reduced below 12 mm², which can be obtained by shrinking the chip by using a smaller feature size. This year most

manufacturers will be offering 0.6 μm technology. Around Q1 99 a number of manufacturers (Motorola, Hitachi, SGS-Thomson, Siemens) will be offering 0.25 - 0.4 μm technology which will reduce the die size with about a factor 3 - 4. This will reduce the die size to an acceptable level.

B.4.4 Product Features

1. **Security Features:** Future generations of integrated circuits will include enhanced security circuits, a memory management unit (MMU). More powerful design methodologies such as high level language design and formal methods that will lead to a higher level of security evaluation and certification. More powerful physical attack protection circuits, memory access controller circuits (bus scrambling and random data on bus while idle) and interwoven layers will provide more protection against hardware attacks.
2. **Low power:** 3.3 and 1.8 V
3. **Increased speed:** increased internal operating frequency (4 times step up)
4. **Cryptography:** Larger word length, true random generator, key generation. For the next generation 32 bit processors it is not clear yet if a cryptographic engine is needed since the CPU has probably enough power to perform the calculation in reasonable time.
5. **Interface:** a mix of contact and contactless interfaces

B.5 Overview

Table 4 gives an overview of the integrated circuits from different vendors that can be used for MAOS systems. The table lists all IC that have at least 10K ROM, 4K EEPROM, 512 bytes of RAM and a cryptographic processor.

Vendor	Device	CPU	ROM	EEPROM	RAM	Crypto Unit	VM	Technology(μm)	Die size (mm ²)	Available
Hitachi	H8/3104		16K	8K	512	-	-	0.8		Now
Hitachi	H8/3103		20K	16K	512	-	-	0.8		Now
Hitachi	H8/3112	8	24K	8K	1321	576	MULTOS3.4	0.8	25	Now
Hitachi	H8/3114	8	32K	16.5 K	1536	1024	MULTOS4.0	0.5	23	Now
Hitachi	H8/3152	8	24K	8.25 K	512	-	-	0.5		Now
Hitachi	H8/3153	8	32K	16.5K	1024	-	-	0.5		Now
Hitachi	H8/3158	8	46K	16.5K	1024	-	-	0.5		Now
Hitachi	H8/3164		48KB	32.5KB	3KB	-		0.35		Q4 99
Atmel-Motorola	MSC0501	8	20K	4K	896	512	-	0.8	18.5	Now
Atmel-Motorola	MSC2114	32	32K	32K	1.6K	1024	-	0.4	22	TBD
Atmel	AT05SC3208C	8	32K	8K	1K	1024		0.5	14	Q300
Atmel	AT05SC3208RF	8	32K	8K	1K			0.5		Q200
Atmel	AT90SC3232C	8 RISC	32K F ³	32K	1K	1024		0.5 / 0.35	24	Now / Q2 00
Atmel	AT90SC1616C	8 RISC	32K F ³	32K	1K	1024		0.5 / 0.35	20	Now / Q2 00
Atmel	AT90SC6464C	8 RISC	64K F ³	64K	2.5K	1024		0.35	25	Q3 00

Vendor	Device	CPU	ROM	EEPROM	RAM	Crypto Unit	VM	Technology(μm)	Die size (mm ²)	Available
NEC	PD789814	8	32K	4K	1K	No	-	0.35		Now
NEC	PD789828	8	32K	8K	1K	Yes	-	0.35		Q2 00
NEC	PD789816	8	32K	16K	1K	No	-	0.35		Q4 99
NEC	PD789819	8	42K	32K	2K	No	-	0.35		Q2 00
NEC	PD789824	8	24K	4K	1K	No	-	0.35		Q4 00
NEC		32	TBD	TBD	TBD	TBD	-			
Philips	P8WE5004	8	32K	4K	1280	1024 /*	-	0.35		2000
Philips	P8WE5008	8	32K	8K	2.3K	1024 /*	-	0.35	6.3	2000
Philips	P8WE5016	8	32K	16K	2304	2048 /*	-	0.35	8.4	Now
Philips	P8WE5032	8	32K	32K	2304	2048 /*	-	0.35	13	Now
Philips	P8WE5033	8	64K	32K	2304	2048 /*	-	0.35		Q2 00
Philips	P8WE6004	8	32K	4K	768	- /*	-	0.35		Q4 99
Philips	P8WE6008	8	32K	8K	768	- /*	-	0.35		2000

Philips	P8WE6016	8	48K	16K	1280	- /*	-	0.35	6.7	Q4 99
Philips	P8WE6032	8	32K	32K	1280	- /*	-	0.35	10	Now
Philips	P8WE6033	8	64K	32K	1280	- /*	-	0.35		Q2 00

Vendor	Device	CPU	ROM	EEPROM	RAM	Crypto Unit	VM	Technology (µm)	Die size (mm ²)	Available
Philips	P16WL032	16	48K	32K	1280	- /*	-	0.35		Q2 00
Philips	P16WA032	16	48K	32K	2.5K	- /*	-	0.35		Q4 99
Philips	P16WA032	16	64K	64K	3K	- /*	-	0.35		Q2 00
SGS-Thomson	ST16CF54B	8	[16K]	4K	512	512	-	0.7		Now
SGS-Thomson	ST19CF68	8	[23K]	8K	1024	512	-	0.6	15	Q2 98
SGS-Thomson	ST19KF16	8	[32K]	16K	2048	1088	-	0.6	15	Q2 99
SGS-Thomson	ST22	32	TBD	TBD	TBD	Yes	JVM, HLL	0.25		Q2 99 ⁺
Infineon	SLE44C10S		7K	1K	256		-			Now
Infineon	SLE44C20S		15K	2K	256+512		-			Now
Infineon	SLE44C24S		24K	2K	256		-	0.6		Now
Infineon	SLE44C42S		15K	4K	256		-	0.6		Now
Infineon	SLE44C80S		15K	8K	256		-	0.6		Now

Infineon	SLE44C84S	24K	8K	256+512		-	0.6	Now
Infineon	SLE44C160S	15K	16K	256		-	0.6	Now
Infineon	SLE44CR42S	14K	4K	256+350	540	-	0.6	Now
Infineon	SLE44CR80S	14K	8K	256+350	540	-	0.6	Now

Vendor	Device	CPU	ROM	EEPROM	RAM	Crypto Unit	VM	Technology (µm)	Die size (mm ²)	Available
Infineon	SLE66C20S		32K	2K	1280		-	0.6		Now
Infineon	SLE66C40S		32K	4K	1280		-	0.6		Now
Infineon	SLE66CL80S		32K	8K	1280		-	0.6		Nov 99
Infineon	SLE66C80S		32K	8K	1280		-	0.6		Now
Infineon	SLE66CX80S		32K	8K	1280	Yes	-	0.6		Now
Infineon	SLE66CL160S		32K	16K	1280		-	0.6		Oct 99
Infineon	SLE66C160S		32K	16K	1280		-	0.6		Now
Infineon	SLE66CX160S		32K	16K	1280	Yes	MULTO S V4 Java	0.6		Now
Infineon	SLE66C320S		32K	32K	1280		-	0.6		Now

Infineon	SLE66CX160P	64K	16K	1280	Yes	-	0.25	Q2 00
Infineon	SLE66CX320P	64K	32K	2048	Yes	-	0.25	Q4 99
Infineon	SLE66CLX320P	64K	32K	2048	Yes	-	0.25	Q2 00
Infineon	SLE66CX640P	64K	64K	4096	Yes	-	0.25	Q4 00
Infineon	SLE88C(L)X320S	64K	32K	4096	Yes	Java MULT OS WinSC	0.25	Q4 00

Vendor	Device	CPU	ROM	EEPROM	RAM	Crypto Unit	VM	Technology (µm)	Die size (mm ²)	Available
Infineon	SLE88CX640S		64K	64K	4096	Yes	Java MULT OS WinSC	0.25		Q2 00
Infineon	SLE88CLX640S		64K	64K	4096	Yes	Java MULT OS WinSC	0.25		Q3 00
Infineon	SLE88CX640S		64K	128K	4096	Yes	Java MULT OS	0.25		2001

							WinSC				
Toshiba	T6N29	8	20K	8K	512	1024	-	0.6	26	Now	
Toshiba	T6N37	8	20K	8K	512	1024	-	0.6	20		
Toshiba	T6N55	8	48K	8K	1024	2048	Yes	0.6	18		
Toshiba	T6N41	8	15K	16K	512	1024	-			99	
Toshiba	T6N42	16	20K	8K	512	2048	-				

Table 4: Overview of IC's suitable for MAOS systems.

* Includes DES-3 coprocessor

[] user ROM, additional part of ROM is used for cryptographic functions

+ Demonstration version