Analysis of J2ME™ for developing Mobile Payment Systems

Master's Thesis in Information Technology,

Electronic Commerce.

IT University of Copenhagen, 01.08.2002

By

Anders Cervera

Email: anders@anderscervera.com

Mobile: +45 26256229

Advisor: Povl Koch

Executive Summary

Mobile commerce has been subject to a lot of hype, among others because of the high market penetration of mobile devices. But no successful mobile payment system has yet lived up the different requirements from the market - and thereby not been a success.

The purpose of this thesis is to describe the factors that affect the introduction of a successful m-payment system - and use these factors to examine whether the J2ME™ technology is suitable for building such successful m-payment systems. J2ME™ is tested, compared with other client technologies and strengths and weaknesses of J2ME™ in relation to m-payments are described.

The conclusions are, that J2ME™ for m-payments depends undeniably on compatibility with the mobile phones and the release of the MIDP 2.0. J2ME™ is suitable for successful m-payments - but depends on the manufacturers willingness to implement specific J2ME™ APIs and facilities in a consistent way. The phones must support push of URLs to ensure smooth download of the payment application and the Wireless API to ensure push and pull of data for easy payment initiating if J2ME™ shall be a part of a successful m-payment system.
Further, the manufacturers must prevent copying of the J2ME™ applications if the phones shall be used as Personal Trusted Devices and support for HTTP over TLS or SSL must be implemented in a consistent way if the phones shall be used as payment terminals. Finally J2ME™ affects the introduction of universal m-payment systems by offering the possibilities of providing a wider range of financial service providers and payment gateways.

Acknowledgements

I wish to thank all the interviewees and organisations for their willingness to contribute to this thesis.

Second I wish to thank the Department for Informatics at Copenhagen Business School for the office and facilities at my disposal during the preparation of this thesis.

Finally the contribution of Povl Koch, Vipul Gupta, Jon Eaves and Hans van der Heijden for their help and valuable comments is very much acknowledged.

Copenhagen, August 2002
Anders Cervera

Abbreviations

2G	The second generation of wireless technology. Usually identified as GSM.
2.5G	Between the second and third generations of wireless technology. Usually identified as GPRS.
3G	Third generation of wireless technology. Usually identified as UMTS.
CLDC	Connected Limited Device Configuration
ECML	Electronic Commerce Modelling Language
EFTPOS	The Electronic Funds Transfer at Point of Sale
FSP	Financial Service Provider
GPRS	General Packet Radio Services
GSM	Global System for Mobile communication
HSCSD	High-Speed Circuit-Switched Data
HTTP	Hyper Text Transfer Protocol
HTTPS	HTTP over SSL (secure socket layer)
IPS	Internet Payment System
JSR	Java Specification Request
KVM	Kilo Virtual Machine. The java virtual machine for mobile devices.
MIDlet	The J2ME™ Application.
MIDlet suite	A bundle of MIDlets in the same application.
MIDP	Mobile information Device Profile
MMS	Multimedia Messaging Service
MPS	M-payment system
NSP	Network Service Provider
PIN	Personal Identification Number
PKI	Public-key infrastructure
POS	Point of sale
PRSMS	Premium Rate SMS System
PSP	Payment service provider
SAT	SIM application Toolkit
SIM	Subscriber Identity Module
SMS	Short Message Service
SMSC	Short Message Server Center
SSL	Secure Socket Layers
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS	Transport Layer Security
UMTS	Universal Mobile Telecommunication Service
WAP	Wireless Application Protocol
WIM	Wireless Identity Module. Usually associated with SIM
WTLS	Wireless Transport Layer Security

TABLE OF CONTENTS

1 Introduction

The introduction of m-commerce hasn't changed the basic rules of business. But the mobility changes the playing field - it offers the possibility to pay for goods and services anywhere and any time.

That is why m-commerce has brought euphoria of excitement and great visions - but some "gaps" are still present, bringing a shadow over the last missing issues to make m-commerce a success.

Many believe, that one of the gaps is missing feasible technologies that help m-payments to live up to a general set of requirements from the market. This thesis examines Java™ 2 Platform, Micro Edition (J2ME™) to see if this is suitable to clear this concrete gap.

Throughout the thesis, I emphasize practical m-payment examples, analysis, tests and comparisons with existing technologies to help clarify the J2ME™ technology. The thesis will describe the strengths and weaknesses of J2ME™ in relation to m-payments and bring a combination of what is suitable now with J2ME™, and what are the future opportunities for further enhancement of J2ME™ for m-payments.

The audience for this thesis is executives, who have dedicated themselves to mobile commerce. This group of people are responsible for business as well as the technology. It includes those who are involved in strategic discussions and those who put the strategic into action - but with a focus on being responsible for providing payment systems and mobile payment systems.

1.1 Problem Formulation

Mobile commerce is commonly expected as part of the new epoch in the digital economy. One of the substantial initiators of e-commerce was the introduction of well functioning and reliable electronic payment systems. It is very likely to suggest that introduction of effective mobile payment systems will increase the interest and general distribution of m-commerce.

There are a several parties in the market, dictating the requirements for mobile payment systems, which is one of the reasons why it is difficult to introduce such systems. The design of mobile payment systems and the use of technology shall somehow meet these requirements.
But it seems that no mobile payment system in Denmark and the Nordic market has met all requirements, which is why no wide acceptance has yet been obtained.

A mobile payment system is typically built of several technologies with all their different possibilities and limitations. There are many different technologies available and new technologies arrive continuously. J2ME™ is a new technology that has recently been introduced at the wireless market and many believe that J2ME™ will improve the diffusion of m-commerce.

As mentioned above, m-payments is one of the fundamental issues for m-commerce why this thesis seeks to answers the following question:

"To what extent is J2ME™ suitable for the client technology in a successful mobile payment system? "

To answer that question, the thesis will initially describe the market, technologies and fundamental aspects of mobile payment systems.

Furthermore the thesis will describe a set of critical success factors (CSFs) and example of requirements. The CSFs shall outline the requirements from the market and describe the challenges of providing a *successful* mobile payment system.

On that basis the thesis will analyse the J2ME™ technology as a building block of a mobile payment system. Comparisons to existing systems and technologies are made during the analysis and the strengths and weaknesses of J2ME™ are described.

The expected conclusion will be an evaluation of the J2ME™ as a building block of a mobile payment system, by describing a set of statements of what eventually needs to be done, to introduce successful m-payment systems with J2ME™.

1.2 Theory

I describe a number of CSFs for m-payment systems and use them as a fundamental step in this thesis to answer my problem formulation. The theoretical framework of CSF is described in [AvFi95] as "a limited number of factors, which are considered critical to the continued success of the business" [AvFi95].

CSF is therefore a framework, which is used to describe the continued success of a business or organisation. An example of a methodology according to the above could be as follows: First, the business goals and objectives are analysed and then the factors critical to achieving each of those objectives are identified. This is followed by an identification of the information and information systems required, if any, to support and monitor these critical CSFs (Top down approach) [AvFi95].

The way I use CSF is basically the same, but just with the purpose of determining the *success* of a *system* instead – i.e. a bottom up approach. I therefore presume that the need for the actual system is- or will be present.

The system is yet new to the market why system improvements are likely needed. According to [AvFi95] an important issue regarding bottom-up approach is to observe existing systems. I therefore also include examples of current systems and technologies to identify their strengths and weaknesses.
Most of the theory used for this purpose is based on interviews, research articles and white papers describing payment systems.

1.3 Methodology and Design

The overall goal of this thesis is to analyse J2ME™ as a building block of an m-payment system. It was necessary to define two subsidiary objectives before this could be done:

1) Overview of m-payments systems and the actors
2) Description of the CSFs of m-payment systems and a set of overall system requirements

Due to an examination of several research methodologies, I found that a Delphi similar survey was an obvious supplemental approach to use according to my thesis. Delphi is a means for aggregating the judgments of a number of individuals in order to improve the quality of decision making [Delbe75].

The subject of this thesis is concerned with quite infant technologies why data is either unavailable or expensive to obtain. The Delphi approach deals with these matters. Normal mail-based surveys were quite too superficial for my purpose. The most obvious method for me was qualitative expert opinions to assist my assumptions and conclusions.

Another important issue was, that time was as a limited factor for me. The strategy for the design of my research should then consider time, limited resources and the

need for expert knowledge and opinions. Inspired by the Delphi approach I decided to collate it with my own designed interview questionnaires.

I designed two carry out to interview rounds with chosen experts. The appropriate number of participants should be between 5 and 10. The participant's chosen were executives who were responsible for advising about or introducing m-payment systems or other kinds of payment systems.
The first round was more open-ended but yet with a beforehand developed set of hypothesises. The next round was more structured to obtain more precise and detailed results.

The aim of the first round where to: Define the parties, technologies and existing systems and discuss the CSFs of m-payments. The aims of the second round where to present the participant with the results and determine the final requirement for m-payment systems.

The duration of the interviews was about 1-1.5 hour and took place at the work of the interviewees.

The following persons were interviewed:

Nature of company	Role of interviewee
Provider of data communication solutions, including mobile roaming, e-business, m-business.	M-commerce manager
Internet payment gateway provider	Manager
Provider of infrastructure and networks for electronic payment processing.	Manger of m-commerce
IT/consultancy provider	Manager of m-commerce
Consultancy agency	Partner + Managing consultant
Mobile phone manufacturer	Prestudy & Conceptual Manager
Mobile phone manufacturer	Technology Manager
Mobile operator	Product Manager M/e Commerce

Table 1: Role of Interviewees and nature of their companies. See Appendix E for an overview of the questions.

The data foundation for the analysis is further supported by extensive desktop research including internet, books, research articles, technical specifications etc.

1.3.1 Thesis Structure

To obtain a solid foundation to address the research question, I first carry out an analysis of the market. The analysis gives an understanding of the following m-payment issues: types of MPSs, definitions, the value-chain, technologies and examples of existing systems.

The next step is to identify the CSFs and requirements for mobile payment systems. Each CSF is summarized by a list of requirements. The requirements give examples of concrete guidelines to implement a successful m-payment system.

After the CSFs and the requirements have been discussed, the J2ME™ can be analysed to identify its suitability as an m-payment building block. This last section walks through the CSFs and evaluates the requirements in relation to J2ME™.

The thesis can be illustrated like this:

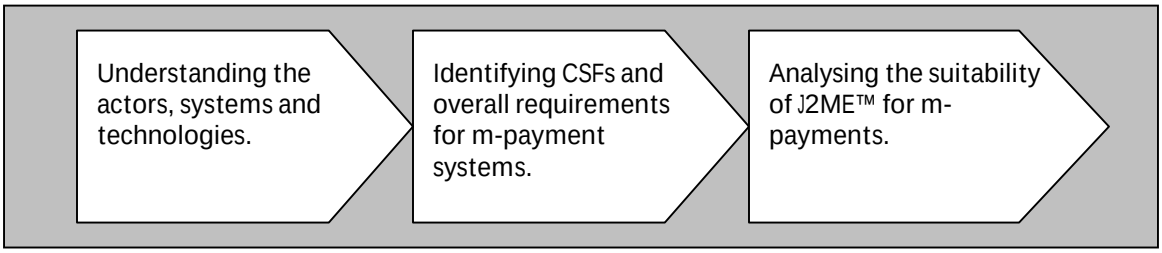

Figure 1: Structure of the thesis.

1.4 Research Scope

An m-payment system consists of several system parts, provided by several parties. Because the core of this thesis concerns J2ME™, my discussion of technologies, CSFs and requirements mainly focuses on the *end-user front end* and the issues regarding data communication to and from the mobile device. Deeper technological issues in relation to the "back end" e.g. payment gateways etc. are therefore not treated.

I will focus on the *mobile phone* as the device for carrying out the payments, even though m-payments can be carried out through several other kinds of devices.

I will also focus on end-user m-payments, i.e. payments carried out by end-users - or B2C payments. Issues about B2B payments and B2B payments systems are not treated in this thesis.

Further basic description of general technologies is avoided. The thesis assumes the readers basic knowledge of technologies hereunder: J2ME™, WAP, SMS, SIM card, GSM, Java, SSL, PKI, Symmetric-key Encryption and Digital Signatures etc.

2 The M-payment Market Place

M-commerce takes place from e-commerce - a fusion of internet technology and wireless applications. Applications based on mobile telecommunication infrastructure provide a new channel for marketing and sales, which in the supply side involves new actors and new technologies.
This first section will describe the m-payment market place by carrying out an analysis of the market actors, systems, technologies and processes in m-payments systems. Further this section will provide a description of tree concrete m-payment systems examples to clarify the new technological challenges of m-payments in the process of data handling to- and from the mobile phone.

2.1 Definitions

Before getting deeper into the subject, a good starting point is to define what a "mobile payment system" (MPS) really is. An appropriate way to define a MPS, is to break the word in the parts "mobile payment" and "system".

First a definition of "mobile payments":

"Mobile payments are payments that are carried out via a mobile device" [Ma01] [wwwit].

Second, the Danish IT encyclopaedia defines a "system" like this:

"A group of related components, which work together, to carry out an assignment" [wwwit].

Now a complete definition of a mobile payment system is possible:

"A group of related components, which work together to carry out payments, that are carried out via a mobile device."

This above definition is to be considered as a pure formal and general definition of a mobile payment system. Later in this thesis, I will of course be more concise about the system and as the definition also implies, an MPS consists of "related components". In the real world, internet payment systems (IPS) or MPS are quite complex and involve several parties with several system parts to integrate with each other.

2.2 Devices

According to the definition above, a mobile payment is carried out via a *mobile device*. A fundamental demand for the mobile device is that it must be able to connect to a network to initiate a payment. The network could be GSM or internet and the clearing and settlement instance could be a bank or mobile operator (explained later in the thesis).

The main qualification to be considered as a mobile device is obviously; that the end-user is able to use it independently of any cables; i.e. the end-user shall be able to carry the device with him/her and be able to use it independently cables for network connections or electric current.
The devices can be one of following representations [CaEY01]:

- GSM-Handsets (mobile phones, two way pagers)
- PocketPC
- Laptops
- PDAs

As mentioned in the research scope I will focus on the mobile phones (a GSM-Handset) as the device carrying out m-payments.

2.3 Why M-payments

Many believe that m-commerce marks the start of a new era in the way of doing business. But m-commerce is just a natural convergence of wireless and standard commerce. As well as internet commerce is an additional way of doing business, m-commerce is likewise an additional way of doing business.

When practicing non-wireless internet commerce, you make the PC the window for you to enter the web, but dependent of you and the PC's location. When practicing m-commerce, you make the web (and all the product or services) come to you, independent of your physical position. You can then buy products or services anywhere and whenever you desire. Using a wireless device as access-device, means that you can increase the available time for business transactions covering any time [KeMa01].

But it's practically of no use to talk about m-commerce, when there's no system to carry out the payment for the product or services you buy. That is why m-payments are one of the critical foundations and key enablers of m-commerce.

But payment is actual a fundamental step in any trade situations, i.e. online via the internet as well as physical POS trade situations.

An interesting aspect about m-payments is that the mobile phone can be used as payment device for all types of payment situations. This is why there are two *basic* forces that indicate a positive future of m-payments [Ma01]+[Dahl01].

- M-payments can be used for all types of payments; e-commerce and standard commerce
- The increasing spread of mobile phones and technology.

M-payments show large potential, among others when considering the new generation (3G) of mobile infrastructure. I will not go deeper into a discussion about which new network infrastructure will be existing in the future, but just stick to the fact that new enhanced network or infrastructure will offer new opportunities of selling content or services. These new opportunities will arise mainly because of two things: Increased bandwidth and the "always on" opportunity. The increased bandwidth offers newer and richer multimedia services and the "always on" opportunity will further the access to the internet and networked services. It is expected that these new services will increase m-commerce and hereby m-payments [Ma01].

The table 2 and 3 below show the expected increase in the spread of mobile phones and the global m-commerce revenues is still large. Note that the numbers marked with a "*" in the table 2 and 3 are forecasts and should therefore be watched carefully, because they are very sensitive to changes in the market.

2000	2002/3*	2010*
570 million	1 billion	1.6 billion

Table 2: Spread of mobile phones, Source: EMC World Cellular Database, [Ma01].
***Forecasts.**

1999	2003*	2010*
3 billion	13 billion	270 billion

Table 3: Global m-commerce revenues. Source: [Ma01], UMTS Forum (2000b), Frost &Sullivan (2000), Barnett/Hodges/Wilshire(2000).
***Forecasts.**

The UMTS Forum estimates that by 2010 half of the mobile subscribers also will be mobile internet subscribers [Umts11]. In 2004, an estimated 350 million people will use mobile ticket purchasing and mobile retail ordering and almost 350 million will use mobile banking and more than 50 million are expected to use mobile financial trading.

By 2005 data traffic is expected to become more important than voice traffic [Ma01]. All mobile services together are expected to generate revenues of about USD 270 billion Euro (including m-payments at the real POS) [Ma01].

2.4 The M-payment Supply Chain

A good starting point to describe the m-payment supply chain is to describe the supply chain for general mobile business.
It is not an exaggeration to argue, that the market for mobile business systems is quite complex. At the time of writing, it is not unusual that there are up to five different parties (excl. the end customer) involved in the process of delivering mobile content to the end customer. But this depends of what kind of product it is, how the product is ordered and how the product is delivered.

The supply-chain for the present mobile business is described in the figure below:

Role	Infra-structure (network providers)	Appli-cation providers	Content-developers/ suppliers	Content - service/ providers	Content aggrega-tors	Mobile portals	Mobile operators	Terminal producers
Example	-Nokia -Ericsson -Lucent	-Realtime -UnWire -Telecom Scandinavia A/S	-Pink Floor -TV2 -Politiken -CNN	-CashU	eWap inMobia	-Nokia Forum -Speedy Tomato	-TDC -Sonofon -Orange	-Motorola -Nokia -Sony/ Ericsson

Table 4: Overview of the supply chain for m-commerce. End-users not included. Inspired by [Futu01].

As illustrated, there are eight possible actors involved in the supply chain of mobile business or services (exclusive the end-users), but not all the players are involved each time an end-user orders a product or service.

An overview of the actors from the market for m-business gives a better foundation to describe the actors in the value chain for m-payments.

An overview of the m-payment supply-chain is illustrated in the table below:

	Regulators					
Role	FSP (Financial service providers	NSP (network service providers)	PSP (payment service providers)	Merchants	Device manufactures	End-users
Example	-Mobile operators -Banks -Ecash providers	-Mobile operators - Other network operators	-Architrade -UnWire - IBM	-Retail shops -Internet shops	-Nokia -Motorola	

Table 5: The supply-chain for m-payments.

There are six main actors involved in a MPS [ShSw98] [Pay01]. The actors can be described as follows:

- **Financial service providers (FSP)**: providing the back end for payment settlement or credit card clearing. There are tree different actors of financial service providers in relation to m-payments. These are *banks*, *mobile operators* and proprietary *electronic cash systems/pre paid accounts* usually provided by the payment service providers (PSP) or FSPs.
- **Payment service providers (PSP)**: They provide the system and access channel to the financial service-provider. Payment service providers establish transactions between the financial service providers and the end-users plus merchants. Basically they provide the software and interfaces between the parties.
- **Merchants (m-service providers):** providing content and/or services to the end-user. The merchants are the ones who sell the goods or services to the end-users. They can sell via internet, mobile phones or at physical locations.
- **End-users:** The end-user buys and pays for the product or service. Providing m-payments is in fact for the sake of the end-users. They are the primary ones who shall accept and adopt the system.
- **Network service Providers (NSP):** providing telecommunication facilities or support infrastructure. Network providers build the network that supports the transmission of payment initiation and verification to and from the mobile device. Following four points can characterize a NSP [BrMc01]:

  - They own the networks.
  - They have the means to identify who is using their network.
  - They have the billing system to charge end-users for their services.
  - They own the relationship with the mobile phone user.
- **Device Manufacturers**: The device manufacturers decide the technological possibilities and limitations of the devices. They design the software infrastructure and the compatibility to 3rd party hardware and software (e.g. J2ME™). Some manufacturers even produce devices with m-payment facilities included in the native software[1].

Another relevant party who indirectly affects m-payments are **regulators**. These are government bodies, law enforcement agencies or banking regulators. They take care of the public interest including protecting end-user rights. These parties affect indirectly by regulating rules and laws in relation to m-payments. A concrete example is the Telecommunication industries association in Denmark [wwwte], a Danish trade association that looks after common interests in the telecommunication sector. They have stated a set of rules in relation to Premium SMS (see existing systems) concerning content restrictions and price regulations.

The six directly involved parties can be divided in two groups: *users* and *system providers*.

- **Users:** These are *end-users* and *merchants*. The users are characterized by that they only use the system and are not directly involved in the system development.
- **System providers:** These are FSPs, PSPs, NSPs or manufacturers. The system providers are characterized by that they are involved in the development and provide the different elements of the MPS.

[1] Se the "example Nokia Wallet".

This distinction of users and system providers are relevant when discussing the CSF and requirements of the systems.

2.4.1 Who is the Financial Service Provider?

An essential question is; who shall take the role as the FSP and provide the settle or clearing service? As mentioned above, there are tree different actors in the category of "financial service providers". These are: **mobile operators**, **banks** or **3rd parties**[2]

The system can be considered as an intermediary between the bank and merchant, if the payment is settled by a 3rd party electronic cash system or pre paid account system. In the end, the end-user account is refilled via transferring funds from the bank to the electronic cash account. These systems primarily arose because of the need for systems that are capable of handling transactions of small amounts (see micro/macro m-payment).

When considering the FSPs without the pre paid systems, the discussion can be summarized in following classifications [Krug01]:

- **The bank dominated model** – where mobile operators only perform data transport to the banks or PSP. The banks are clearing the transactions.
- **Mobile operator dominated model** – where mobile operators performs the billing via their existent billing system. The banks are not involved.

Mobile operators are to some extent obvious candidates to take the role as PSP in an MPS. Mobile operators are all ready billing the same customers, they own the network and have the technical expertise. But a number of problems arise if mobile operators are being in charge of the settlement for 3rd party products. The mobile operator is burdened with the additional financial risk when becoming a bank and must therefore acquire a banking or EMI license. The EMI directive aims to provide a

[2] 3rd parties are e.g. gateways with own proprietary pre paid account systems or e-cash systems.

simplified regulatory framework for institutions that want to provide payment system based on electronic money.
Another disadvantage of mobile operators being the FSP is that the payment system is probably limited to the current subscribers (see closed systems below).

The bank dominated model includes retail banks or Card Issuers like VISA, MasterCard or store cards/loyalty cards. The bank dominated model is also appropriate because clearing and settlement is already a part of their cores business. They do have a trusted brand and experience in relation to risk management and they have the regulatory approval to take care of the end-users money [BrMc01].

But there are also obvious reasons for both parties to offer financial services in a MPS - these are that they both all ready are providing billing services and have a significant existing customer base.

2.5 M-payment distinctions

2.5.1 Open vs. Closed M-payment Systems

M-payment systems can be divided in two categories depending on the FSP (bank or mobile operator); *Open* MPSs and *closed* MPSs.
In the closed MPS the customer and the merchant need to have a contract with the same clearinghouse. The open MPS allows customers to pay regardless of which FSP he is contracted to.

Supporting different kinds of m-payments makes the MPS more universal. The MPS is more flexible and covers a broader range of need from the end-users, if the MPS e.g. supports micro or macro m-payments (see below).

2.5.2 Micro/Macro M-payment

When describing MPSs, it is furthermore relevant to distinguish if the product is *electronically* or *physically* or if the current system is handling *micro m-payments* or *macro m-payments*.

The figure below illustrates the different possibilities in which m-payments can be categorized:

	Micro m-payment	**Macro m-payment**
Electronic goods	E.g. news, seeking, alerts	E.g. stocks, movies
Physical goods	E.g. parking meters, ticket dispensers.	E.g. super market, retail

Table 6: Different m-payment scenarios.

As mentioned above, micro or macro m-payments refer to the size of the amount being transferred.
Micro payment can be defined as a "low value electronic financial transactions" [wwww3] and refers to payments systems that are capable to transfer very small amounts. Most of the business models concerning electronic goods are dealing with marginal profits and very small prices.
There are today two ways to clear micro m-payments in Denmark: via Premium Rate SMS or via mobile interface to prepaid electronic cash accounts.
The banks, PBS or mobile operators does not directly support any micro-payment system at the present [3]. The way the banks and mobile operator have worked around this issue is by establishing a joint venture company, which provides a system that supports micro-payments [wwwco].

Macro payments are payments that exceed the maximum value accepted in micro payment systems and are therefore mostly carried out through other payment

[3] Technically the PBS system does support micro-payments.

systems. Macro payments are typically carried out via financial institutions that have achieved a licence to carry out banking business.

2.5.3 Types of Products

There are in general two kinds of products when performing m-commerce: physical products and digital products. The digital products are mostly delivered to the mobile device, but could also be delivered via a web page or an email. Electronic products or services can be consumed via a complete (downloadable) application or via network i.e. voice, WAP, SMS etc.

The digital products, also mentioned as content, can be listed in following categories:

Services:

- Messaging services (Notification etc.)
- Directory enquiries (phone numbers, addresses etc.)
- Commercial content (News etc.)
- Offline applications (Dictionary, road maps)

Entertainment:

- Messaging content (Voice greetings, jokes, horoscopes etc.,)
- Online games (multiple choice etc.)
- Online services (Chat, dating)
- Offline games (Pac man, space invaders etc.)

The selection of the physical products is at the moment very delimited, but could in reality be every imaginable product. There are only few examples of physical products in Denmark, which can be paid via the mobile phone. The available examples are parking tickets and Coca Cola vendor machines. But after the newly introduced m-payment system "mPay" (see: "2.7.3 example: mPay") it is now possible for retail merchants to sign up and offer physical products.

2.6 M-payment Standards

2.6.1 Forums/Organisations

As mentioned earlier m-payments are a new alternative to the existing payment/billing systems. M-payments are using existing backend systems for clearing and settlement and therefore use existing standards for network protocols and bearers.

But some attempts to describe concrete m-payment protocols and languages have already been carried out. Like other initiatives in relation to the internet, the m-payment issue have caused a number of forums to debate-, accelerate and influence the development of universal global standards for interoperability.

A number of the most noteworthy forums are:

- **Mobile Payment Forum**, HTTP://www.mobilepaymentforum.org. Mobile Payment Forum creates a framework for mobile commerce using payment card accounts.
- **Mobile electronic Transactions (Met)**, HTTP://www.mobiletransaction.org. MeT focuses on aspects of digital signatures and PKI for mobile devices.
- **Mobey Forum** (Mobile Financial Services), HTTP://www.mobeyforum.org. Mobey focuses on how to use the end-user mobile phone as a personal and trusted device. This addresses security issues for mobile execution of financial services: payment, remote banking and brokerage.
- **PayCircle** (Payment group), HTTP://www.paycircle.org. PayCircle addresses a business-enabling infrastructure for web-services with the focus on micro payment.
- **Radicchio** - HTTP://www.radicchio.org. Radicchio promotes digital signature and PKI for the mobile environment.
- **GMCF** - HTTP://www.gmcforum.com/mainset.html. GMCIG enables security and interoperability for mobile macro payments.

- **Open Mobile Alliance (OMA) -** http://www.openmobilealliance.org (former WAP forum). OMA focuses on the entire mobile industry hereunder the standards and specifications in relation to WAP.

An interesting issue about the forums is, that most of the members are from across industries (across the supply chain), with the common goal to accelerate m-payments.

A concrete example of an m-payment standard, which is a result from a forum co-operation, is *ECML*[4]. ECML (Electronic Commerce Modelling Language) is e.g. used in the Nokia Wallet[5] for making purchases around the Web.
Digital wallets are software tools that allow end-users to store billing, shipping and payment information and use this information to complete a merchant's checkout page automatically. They may be used as browser plug-ins or helper applications, stand-alone client applications or server-based applications.

The proliferation of digital wallets, however, has been hampered by the lack of standards. Use of ECML by multiple vendors of digital wallets would mean that Web shoppers could drag and drop a computer icon to make a purchase from any participating merchant [ITUecml99].

2.6.2 Protocols and Technologies for Connectivity

M-payment data are just like any other data, why no new special network standard is needed to carry out m-payment transactions. M-payments are therefore carried out through existing networks, which could be e.g.: Wireless LAN (IEEE 802.11 protocol), Bluetooth, Infrared (irDa) and Cellular networks (GSM/2,5G/3G).

[4] The standard was initially started by the "ECML Alliance" but now control and development has been transferred to IETF [PaEa02].

[5] Se the "example: Nokia Wallet".

But new protocols and technologies for connectivity have been created in relation to m-payments. Two new challenges arrive when adding the mobile phone as new payment terminal - these are:

- *Data handling* and *presentation* in the mobile phone
- *Data transmission* to and from the mobile phone

The following protocols and technologies listed below facilitate the different methods/techniques of *handling* and *transmitting* data to and from the mobile phone, to carry out an m-payment transaction. The list is not complete, but the most important technologies for m-payment connectivity are listed:

SIM Application Toolkit (SAT)

SAT is a technology that allows configuring and programming the SIM card[6] [gempl01] [GuCr02]. The SIM card contains simple application logic that is able to exchange data with the SMSC, to carry out m-payment transactions. The specific mobile operator provides the application logic and is responsible of providing the SIM card.

The bearer of the communication between the mobile device (SAT) and SMSC is **SMS** (e.g. example: mPay) and the network is e.g. GSM, 2,5G or 3G.

WAP/WTLS/WIM

Phones equipped with a WAP-browser are able to exchange data with any web-server. Data is transmitted via wireless application protocol and the networks are e.g. GSM, 2.5G or 3G. The WAP 2.0 stack supports services on TCP/IP, TLS and HTTP [Wap2.0].

[6] Requires that the SIM card supports SAT.

WTLS is a layer in the WAP stack and is the wireless edition of the SSL 3.0 in a reduced scale. WTLS can provide secure connections for transferring confidential data [WTLS02].

WIM is a module for storing data in the mobile device and is usually used in relation to WAP transactions. WIM is e.g. used with WTLS transaction to protect permanent, typically certified, private keys. The WIM stores these keys and performs operation using these keys [WIM01].

SMS

The device can exchange data via a SMSC by sending and receiving standard SMS messages. By using SMS to initiate or authorize payments the SMS can be the used as the unit of currency itself (see example: Premium rate SMS)[7].

Voice

The end-user can via a normal phone call state his credit card number to the merchant who transfers the funds via interface provided by a PSP. A voice response system at the payment service provider can also call the end-user and guide him through a payment procedure. Voice recognition can also be used as an authentication tool for payment settlement.

Manufacturer specific Applications

The mobile phone manufacturers can chose to install native applications, which e.g. in interaction with one of the above technologies enables m-payment opportunities.

[7] The most frequent used protocols to communicate with the SMSCs are CIMD and CIMD2 (Nokia), UCP/EMI (CMG), OIS 5.4 (SEMA), SMPP 3.3 and 3.4 (Aldiscon, Comverse, Ericsson, Unisys, SEMA) [UnWi02].

2.7 M-payment Systems

This section describes the foundational parts in an MPS and gives an overview of how the system works. Further this section describes examples of concrete MPSs, where the above elements are constituent parts related to the mobile device.

2.7.1 A general M-payment System

A MPS has a front-end and a back-end like other client server systems. But because the system has two different kinds of users, the front-end is divided into two: a front end for merchants and a front end for the end-users.

The front end for the end-users is the software/application running on the mobile device and the back-end is processing the payment request and settlement.
In a simplified m-commerce scenario there are tree parties interacting with the MPS: the *end-user*, the *merchant* and the *FSP*.

An abstract overview of the simplified MPS can be illustrated like this:

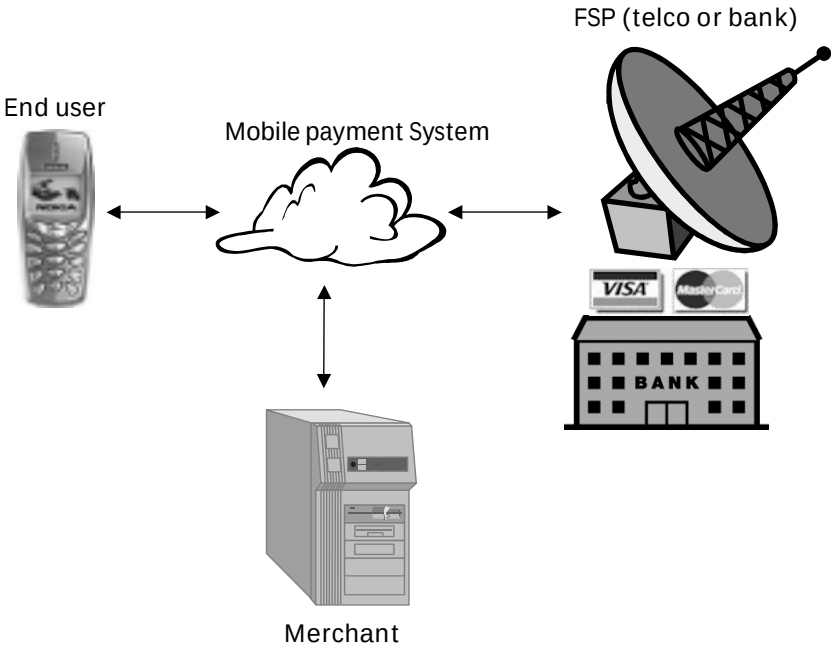

Figure 2: High-level abstract overview of a mobile payment system.

The above illustration is very simplified and contains no intermediaries - i.e. PSPs. Real life system usually involves PSPs to handle settlement or even content delivery because merchants usually doesn't obtain the resources and capital to build and maintain such systems.

Let's get deeper into the flow of the system. A "standard" flow of a buy and payment via the internet usually consists of following steps between the parties:

Action	Parties
1) Purchase initiation	End-user ⇔ Merchant
2) User and/or Account identification	End-user ⇔ PSP (FSP*)
3) Authorize payment	PSP (FSP*) ⇔ FSP
4) Payment authorized	FPS ⇔ Merchant
5) Recipient + content	Merchant ⇔ End-user
6) Payment capture**	Merchant ⇔ FSP**

Table 7: Standard procedure for an internet payment.
*** If no PSP are invovled.**
****According to the Danish law the Merchant is only allowed to transfer the funds (capture) after the products are shipped.**

But the above basic structure of the payment flow is actually not affected whether the end-user buys via the internet or via a mobile device.

The actual difference between the MPS and a standard IPS is the interface to the end-user - how to exchange data between the mobile device and backend. And when examining step 2 in the above table, the interface to the end-user handles a very critical step, namely: *identification* of the customer or transmission of *confidential credit card* information. Like any other payment system a significant challenge is to ether encrypt confidential data or identification of the end-user (see CSFs).

2.7.2 Example: Premium rate SMS

Jokes, ring tones, operator logos or SMS alerts delivered to the mobile phone was among the first successful mobile services in Scandinavia. These services have all one thing in common: they are *delivered* via SMS directly to the phone and also *paid* via the SMS - that is because SMS is an obvious way to charge for these services. This type of MPS is "Premium rate SMS" (prsms).

Prsms is at the present a widespread MPS in the Nordic. But prsms is actually not a billing technology in itself but more like a special technique to charge for content. Prsms is when the mobile operator (FSP) charges the customer an excess price for the SMS, because the customer also receives content within (or before) the specific SMS. The revenue generated from the excess price, is then shared between the content provider and the mobile operators.

An simplified example of a how the flow takes steps when using a prsms system is illustrated in the figure below:

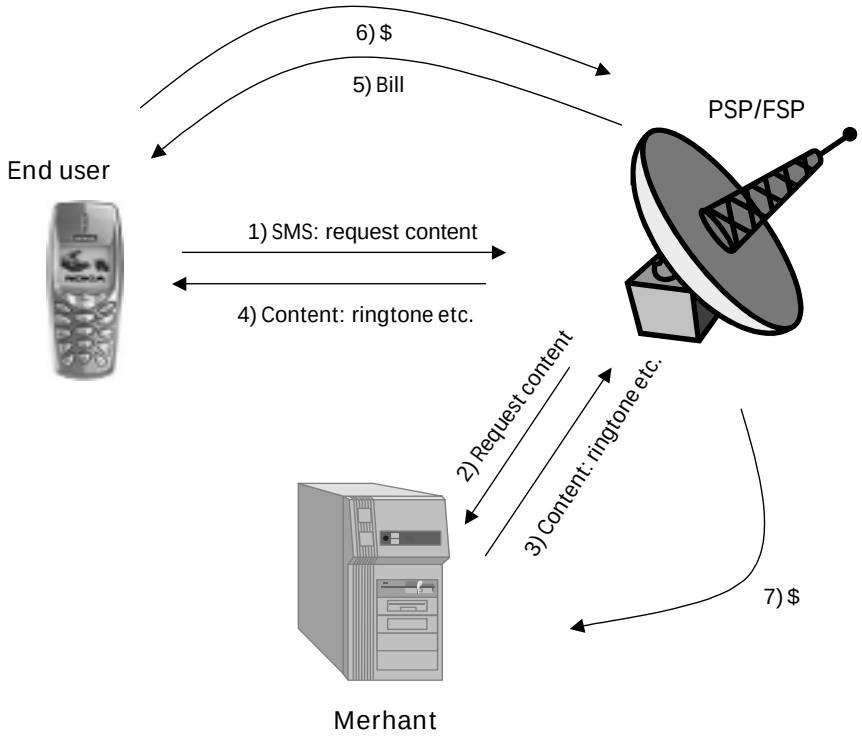

Figure 3: Example of a high level flow when using a prsms system.

1) The user requests a service by entering a key word in the SMS and sends it to the PSP/FSP (mobile operator or 3rd party).
2) The PSP/FSP forwards the request e.g. via HTTP to the merchant.
3) The merchant returns the service/content to PSP/FSP.
4) The PSP/FSP forwards the service/content to the end-user in a prsms.

5/6/7) The PSP/FSP bills the end-user and credits the merchant.

No confidential data is transmitted or stored in the mobile phone. The phone is just used as an "authenticator" to identify the user and accept the payment. The identification is done via the phone number, which is transmitted inside the SMS.

The above figure shows an example of the flow when a customer buys e.g. a ring tone to his mobile phone. In this example the figure is simplified. Usually there is an PSP (prsms gateway) between the mobile operator and merchant to handle the SMS request and delivering content to the mobile device.

The technique of charging via SMS is a bit more complicated than e.g. charging via the internet. The PSP needs a SMS gateway to send the specific prsms and the gateway need to be interoperable with all the mobile operators[8]. In the major part of the prsms systems, the PSP provides an HTTP interface for the merchant to the SMS gateway. The merchant can use the interface to initiate payments i.e. send a premium SMS to the end-users [UnWi02].

2.7.3 Example: mPay, SIM Tool Kit (SAT)

"Orange Mobil Betaling" or mPay is an MPS developed in corporation between PBS, Orange and Gemplus. The system is based on a SAT application.
To use this system, the customer is required initially to complete an application at Orange[9]. The customer has to fill out his phone number, specific pin-code and credit card informations. When engaged in the system, the customer is able to order products via WAP, voice or an internet browser – and pay via the mobile phone.

[8] I.e. if subscribers from all mobile operators shall be able to use the system.

[9] At this moment, Orange is the only mobile operator providing this particular system.

The system flow can be illustrated like this:

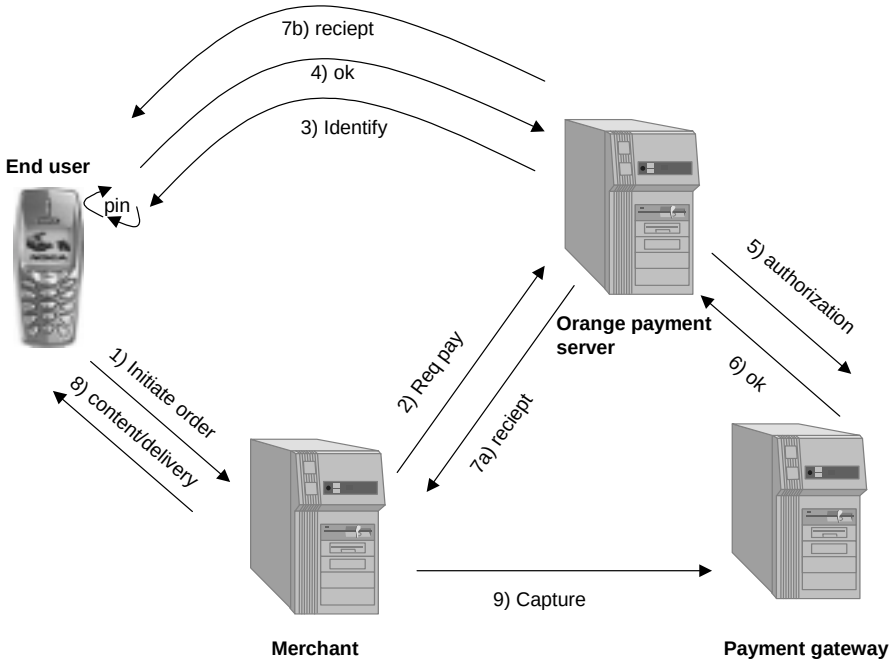

Figure 4: mPay, high-level overview [PBSsmps].

1) The end-user has to give up his phone number to the merchant, either via WAP, sms or the internet when he/her shall pay.
2) The merchant then transfers the order request to a payment server at Orange.
3+4) Orange then sends the order information to the mobile phone, which the end-user has to confirm and authorize by pin code.
5+6) The confirmation is then send back to Orange' payment server. The payment server transfers the request to the PBS payment gateway that authorizes the clearing.
7+8) A receipt is generated to the merchant who sends the product/service to the end-user.
9) After successful delivery the merchant debits his money.

No confidential data is transmitted or stored in the mobile phone. The phone is just used as an "authenticator" to identify the user and accept the payment.
This system uses two basic techniques to ensure secure m-payment:

- End-to-end encryption between the mobile device and the payment gateway.
- PIN to identify the correct user and mobile phone.

The SIM card and PBS payment gateway contains the corresponding cryptographic keys to enable encrypted transmission between device and server. For further authentication the user has to enter a unique PIN code, which is also stored in the SIM. The PIN is entered and verified off-line by the SIM application and a authorization message is send, based upon a triple DES message digest, which implicitly proves that the correct PIN has been entered [PBSsmps].

The mPay system is based on standard back end systems, why all payment authorization and settlement are handled by existing systems at PBS. The merchant just has to interface with an existing PBS's payment gateway (e.g. via the PSIP specifications [PbsPsip]) and add some extra values in the requests.

2.7.4 Example: Nokia wallet (WAP)

The wallet application enables end-users to make payment transactions via a WAP-browser, by using stored personal information such as credit card details inside the mobile phone [NoWal]. The wallet application makes it easier to pay via WAP by automatically fetching credit card information from the wallet and fills in the required fields in the WAP browser. The data stored in the wallet is protected with a PIN code to authenticate the end-user to the application.
The wallet also supports the use of WTLS (for server authentication and data encryption) or optional WIM containing digital signatures. The wallet application is based on WAP as bearer and ECML (Electronic Commerce Modelling Language) as data standard.

The Nokia Wallet is at the present supported by Nokia 6310, 6310i and 6510 phones.

2.8 The M-payment Client

If you compare the mobile phone with standard payments e.g. EFPOS, the most obvious roles for the mobile phones are: as a *terminal* or *personal trusted device* (PTD)[10]. The PTD can further act as either a *wallet* or *authenticator*.

The different roles can be describes as follows:

- **Terminal** (e.g. the WAP/WTLS technology): The mobile phone only transmits credit card numbers or other accounting data to use for clearing and settlement. No personal data is entered to identify the end-user or who is paying. The terminal is acting similar to an EFTPOS.
- **Wallet, PTD** (e.g. Nokia wallet): The phone stores the credit card numbers and transmits them when paying. The end-user "opens" the wallet (authenticates) by e.g. entering a PIN.
- **Authenticator, PTD** (mPay example): No confidential data (e.g. credit card numbers) is necessarily transmitted. The data to carry out the clearing and settlement is known by the PSP or FSP. Thereby the end-user only *accepts* the payment on the phone by e.g. entering PIN or carry out other procedure to authentication. The authentication is equal to accepting the actual payment. This type of payment is typically used in *mobile operator dominated m-payment systems* where no credit card numbers are needed. The mobile operators identifies the accounting data e.g. via the number of the mobile phone.

[10] According to the MeT-forum a PTD contains the following aspects:

1) It is personal, controlled and use by one person and carried by that person most of the time.

2) It has an application platform with associated user interfaces for transaction related services such as banking, payment, and bonus programs.

3) It has security functionality required for transaction related services: secure sessions, authentication and authorization [MetPTD].

Furthermore the technology for connectivity in the mobile phone can present content in two different ways:

- **Browser**: The merchant is deciding most of the application flow, by presenting products in e.g. WML/XML pages from a web server via a browser in the mobile phone.
- **Static application logic**: The application is running on the mobile phone independent of any web server. The system providers determine the flow and application logic.

2.9 Summary

M-payments are predicted glorious future basically because of two things: m-payments can be used for *all types of payments* - anywhere and any time, and because of the *increasing spread of mobile phones*.

M-payments can theoretically be carried out via *any wireless device*, but the most obvious devices are mobile phones, PocketPCs, Laptops and PDAs.

There are six important actors involved in the value chain for MPSs. The actors can be divided in *users* and *system providers* - the users are *end-users* and *merchants*, and the system providers can be NSPs, FSPs, PSPs and manufacturers.
The FSPs can be divided in *banks* or *mobile operators*, where the *bank-dominated model* describes the banks as FPS and the *mobile operator-dominated model* describes the mobile operators as the FSP.

MPSs can be *open* or *closed* systems. Open systems, also referred to as *universal* payment systems, are where end-users are allowed to pay regardless of the kind of FSP. M-payments can also be divided into *micro* or *macro* m-payments and the product or services being paid for can be *physical* or *electronically*.

A lot of initiatives have been established solely in respect to m-payments. The members are from across industries and they all intend to develop standard and universal MPS.

There are two important challenges, when focusing on the technology for device connectivity - these are *data handling* and *presenting* in the mobile phone and *data transmission* to- and from the mobile phone. Several technologies are available, but the most obvious are: *SAT, WAP, SMS, VOICE* and *manufacturer specific* applications.

The mobile phone (or client) in a MPS can act different roles. The mobile phone can act as a *terminal* or a *PTD*. The PTD can be a *wallet* or an *authenticator*.
The terminal is similar to an EFTPOS and the PTD is storing the confidential data like PIN for *authentication* or credit card number for automatic transmission.
Finally the application logic for the client is either stored in the phone or placed at a server.

The above description of the actors, technologies, examples and definitions, will now be used as the basic guideline and foundation when describing the most important factors to make the MPS a success.

3 Critical Success Factors of M-payment Systems

No standardized and generally adopted MPS has yet emerged in Denmark or in the EU. One reason could be, that the existing systems somehow not live up to all the requirements and demands from the market actors.
This section lines up the critical success factors (CSF) that affect the successful introduction of an MPS. Each CSF is briefly explained and related to concrete examples for clearer understanding. A set of concrete *requirements* is defined after each CSF. Note that some of the requirements may be difficult to fulfil completely at the present, but must be considered as examples of an ideal m-payment system.

As described earlier an MPS can be considered as a network of six participants: manufacturers, merchants, end-users, PSPs, FSPs and NSPs. These are further divided in users and system providers. There are different CSFs and requirements to the system because of the involvement of different actors.

According to the theory of positive feedback and network externalities [ShVa99] the perceived value from the participants increases every time the network grows larger. E.g. each time a new merchant enters the network and offers m-payments, the perceived value of the end-users increases because the supply of products is getting bigger. And each time a end-user enters the network and is capable of paying via their mobile phones, the perceived value of the merchant increases because his market gets bigger. And additional mobile operators or banks in the network mean extra end-users, and so on.

That is why an important means of getting a successful MPS, is obtaining acceptance from all the participants in the network and thereby achieving a critical mass.

By the result of the study, [ShSw98]+[Heij02] and the two rounds of interviews, the following CSFs are identified:

Factors	Related factors
Ease of use	Intuitive, few clicks, flexibility, unobtrusiveness, performance, installing/download
Expenses	Transaction fees, set up fees, subscription fees
Security	Reliability, privacy, anonymity, trustworthiness, regulatory framework, regulation, end-user protection, confidentiality, Integrity, authentication, Verification / Non-repudiation
Universality	Critical mass, transferability, divisibility, standardization.
Technical Feasibility	Integration effort, interoperability, scalability, remote access, performance

Table 8: Critical success factors of m-payment systems and their related factors ([Heij02], [ShSw98]).

Each CSF is explained separately in the following. After each CSF a set of concrete requirements are derived.

3.1 Ease of use

From the *users* (end-users/merchants) point of view, a new payment system some how has to "compete" with the existing ways end-users normally carry out payments. Today the two major payments methods in retail business in Denmark are cash paid at POS and credit/debit card terminal, EFTPOS. The most widespread payment system on Danish websites is SSL transactions with Visa/Dankort [wwwbe].

The advantage for the above-mentioned two existing physical payment method is, that they are relatively *easy to use*.

Lets look at some examples of what could be potential sinners when considering the term "ease of use". The "ease of use" can be related to the process *initially being able* to use the system, activities *during* the transaction or *administering* the system.

Looking at existing pure cash payment there are no additional activities or processes related to the POS cash transaction or EFTPOS. There are some minor activities related to a SSL transaction via the internet or EFTPOS such as entering credit card numbers or an pin-code.
The following list is examples of concrete potential concerns in relation to "ease of use". The concerns are categorized upon end-users and merchants:

End-users:

- Initial "set up" procedure (e.g. filling out forms, corresponding with banks etc.)
- Initial download and installing of software (or upgrade)
- Usability: Intuitive, few clicks, performance, easy entering

Merchants:

- Carrying out payment capture
- Checking statistics
- Carrying out refunds

A concrete example of an IPS that has not succeeded yet is the SET certificate. To be able to use the SET certificate, the end-user has to go through following procedure [wwwbe]:

- Print a formula from the internet and sending it in a physical letter to a bank.
- Receive the SET software in physical mail and install it.
- Receive a one-off pin-code via physical mail to acquire the certificate.
- Download and install two certificates; to Dankort and Visa Card.

Several factors indicates that SET is more secure that SSL [wwwbe], but the customers and merchants has obviously chosen SSL for the fact that it is easier to

use, and it is no less un-secure than using their credit/debit card else where in the physical world. Another point in this example is, that the merchants were not very accommodated to use and spread out SET either - which results in negative feedback and no network effect.

There's no initial set up procedure to be able to pay via SSL on the internet. When using SSL to pay via the internet the user only has to enter the credit card number (incl. a check number). This advantage may not be disregarded if the users shall generally accept a new m-payment system - The new m-payment system should therefore be at least as easy to use.

Examples of Requirements

The burden for the end-user or merchant to be set-up or engaged in the system *must be minimal*.
The initial set-up procedure is not dependent of the technology itself but how the overall system is designed and the *role* of the mobile phone. If the phone has to "act" as an *authenticator*, a set-up procedure is inevitable for the end-user. If the phone shall act as a pure a *wallet* or *terminal,* a set-up procedure for the end-user is avoidable - dependent of what technology is used.
If the system relies on existing platforms from FSP and PSP, the set-up procedure is minimal for the merchant.
As a principal of rule, there may *not be any installation procedure* above a simple acceptance of the concrete application download.
Another important rule is, that if the system has to be "at least as easy" as existing payment methods like e.g. EFTPOS, the system *must be intuitive* and the *quantity of clicks and user typing must be absolute minimal*. A key in of URL may not be acceptable. A key in of phone number in a SMS may be acceptable.

3.2 Expenses

Expenses are a critical susses factor, which mainly affect the *end-users*, *merchants* and the *system providers*. In this case the system providers could be PSPs, FSP or NSPs.
As described earlier, a MPS can be considered as a supplement and is in this case not a *necessity* for the merchants and end-users. Because MPSs is considered as an extra supplement, eventual extra expenditures are sensitive and may be compared to expenses from existing payment systems.

The following types of expenditures applies to existing internet payment systems [wwwbe]:

For Merchants:

- Initial set-up fee (POS terminals, Software)
- Monthly fee
- Transaction fee
- Certification fee [only via PBS]
- Charge back fee [only via PBS]
- Risk fee [only via PBS]

For End-users:

- No direct extra fees are placed on the end-users to carrying out general payments.

When adding the mobile dimension to a payment system, additional expenditures may be added - these are:

For merchants:

- The same standard transaction fees as for existing payment systems. Though in most cases an *initial fee* is charged for setting up the mobile integration[11].

For end-users:

[11] To provide mPay or prsms in Denmark an initial fee is charged.

- SMS fee (for prsms, or SAT based systems)
- Standard rate for voice calling (voice response based systems)
- Upgrading the SIM card [12]
- Airtime (data transmission)

For System Providers:

- Developing costs
- Deployment costs (e.g. web-hosting etc.)
- System maintenance
- Upgrading the SIM card (If the system depends on SIM, an upgrade could cause the need to provide new SIM cards)

Existing payment systems do not provide any additional fees for the end-user, why it is likely to suggest, that MPS must bring no initial expenditures or additional transaction cost for the end-users.

For the merchant point of view, the system shall be profitable compared to existing systems and therefore be able to carry out low priced transactions and system maintenance.

The expenses held by the system providers are for building and providing the systems for the users. These expenses must in this situation bear comparison with the concrete business case. The development costs are to some degree likely imposed on the users – but again, this depends on the concrete business case.

[12] If consumers are using older SIM cards, which are not compatible with SAT, the SIM card has to be exchanged. The price in Denmark for a SIM Toolkit 2+ are approx DKK 250,-

Examples of requirements

A new MPS must be *free of charge* for the end-users. Further *no direct transaction cost must be charged*. Indirect cost like e.g. airtime or transmission of SMS may be acceptable if it is kept at an absolutely minimum.
A minor set-up fee may be acceptable for the merchant, but *extra transaction fees exceeding the standard fees for e.g. internet transactions may not be acceptable*.
The development cost for the system providers must *be reasonable* in relation to the actual business case.

3.3 Security

The interviewees stressed the security as a very *fundamental* CSF in relation to IPS and MPS. If the security is low or unacceptable, the system will never be introduced and the other CSFs make no difference.

Security is also a CSF that *all of the involved parties* can affect.
The end-users shall watch out when entering PIN or other confidential information and pay attention not to lose the mobile device. The Merchant, PSP and FSP must secure their individual servers, databases and network connections. And the system must be designed to transfer data in a secure way between all interfaces.

There are a number of factors, which can be sub-categorized below security.
According to [Heij02] the security can be categorized into following related factors:

- Reliability
- Privacy
- Anonymity
- Trustworthiness
- Regulatory framework
- Regulation
- End-user protection.

The interviewees also mentioned these factors, but were described as a set of overall factors mostly related to the *back-end*. Furthermore they are not in particular related to m-payment systems but payment systems in general.

But new critical security challenge appears when the mobile device is added as a new front-end payment unit. The new security issue is concerning the *data handling* in the mobile phone and the *data transmission* between the back-end and the mobile phone.

But the CSFs for protecting m-payment transactions according to the end-user front-end are actually similar to the requirements for protecting standard payment transactions; Sensitive data from a *trusted end-user* must be *secured throughout the transmission*.

But in this case the mobile phone has become a *payment terminal* or a *wallet*. New hardware, network and protocols have to be integrated in the payment system, which complicates the security a great deal.

The interviewees agreed on the following new significant security challenges:

- Confidentiality
- Integrity
- Authentication
- Verification / Non-repudiation

Confidentiality prevents *eavesdropping* i.e. that the information remains intact, but its privacy is compromised. For example if someone "listens" on the network traffic and copies the credit card number [NeInPu].

Integrity prevents *tampering* i.e. the data is changed or replaced before it has reached the recipient. For example someone could alter the amount or merchant id of the payment [NeInPu].

Authentication prevents *impersonation*. Authentication allows the recipient of information to determine its origin – that is, to confirm the sender's identity [NeInPu]. Authentication can further be divided in server- and client authentication.

Verification prevents *non-repudiation*. Verification allows the end-user to approve the transaction data before carrying out the transaction. The transaction data summarises e.g. the amount, goods and merchant. The end-user digitally "signs" and accepts the data when confirming the data. A key e.g. signs the digital signature and a hash or MAC value of the text is also generated.
Non-repudiation means a prove to the merchant and/or FSP that the end-user has confirmed the "actual" data that was presented at the mobile phone, and the integrity of the data

There are many different technologies to comply with the above CSFs, but the most obvious techniques mentioned by the interviewees can be listed in the following table:

	Techniques	Examples
Confidentiality	-Public key encryption -Symmetric key encryption	- SSL protocol - RC4 chiper - Triple DES chiper
Integrity	-One way hash	- SHA-1 digest
Authentication	-Certificate -Symmetric key -PIN -Biometrics	- X509 v3, certificate - SSL protocol
Verification / non-repudiation	-Digital signature -MAC	- Digest + symmetric key.

Table 9: Security, techniques and examples

Important techniques for confidentiality are public- or symmetric key encryption. Symmetric key ciphers provides stronger encryption that public-key encryption ciphers [NeInPu]. This means, that symmetric-key encryption needs proportional smaller keys that public-key encryption does. In addition the symmetric key encryption is much faster than the public-key encryption, but public-key encryption provides better authentication techniques (the digital signature) [NeInSS].
This could have an influence in relation to the processor power used for encryption, which is an important issue in mobile devices with limited processor resources[13].

PIN code authorization if for authenticate the user of the mobile device. This is an important issue, if the mobile device is carrying personal and confidential information. But either PIN-, symmetric key or certificate-based authentication is related to a 3rd party *physical access* to the mobile phone. The situation is almost the

[13] I have tested the performance of J2ME™ in relation to security by measuring the speed of selected ciphers for encryption. See the test results in the section "4.2.5 J2ME™ and technical feasibility".

same as having the wallet stolen, if 3rd party is aware of the PIN protecting access to credit card number.

Finally an important problem with symmetric key based m-payments is getting the sender and receiver to agree on the key without the fear of eavesdropping [NelPKI].

Example of Requirements

If the phone is used as a *wallet* or *authenticator*, it must support user *authenticating* to hinder abuse if the phone/wallet is lost. The authentication could be PIN or symmetric key stored in SIM/WIM or other persistent storage on the phone.
The system must support *verification/non-repudiation* if the merchants or FSP requires proof for the transactions - e.g. a digitally signed piece of data. The data is then signed by the end-user, transmitted with a HASH digest or as a MAC value to ensure *integrity*. The verification can be stored as a proof for the actual transaction (*non-repudiation*).

The phones must support *cryptology* to secure confidentiality if the phone is used as a *terminal*. The cryptology must take place from "*end to end*" i.e. from the device to the backend system that is processing the payment (PSP or FSP). Some systems are dependent of a "payment gateway" (mostly provided by a PSP), which could make it difficult for enabling pure end-to-end encryption. This depends of the technologies used and the overall system design (This problems is further discussed in section of J2ME™ and security).

As explained above, there are different types and strengths of cryptology. An obvious cryptology method is the SSL protocol. Les advanced methods like pure symmetric key could also do the work of cryptology.

3.4 Technical Feasibility

Technical feasibility relates to the technology being used and the design of the actual system. The underlying related factors to technical feasibility are:

- Integration effort
- Interoperability
- Scalability
- Remote access
- Performance

As mentioned by Hans van der Heijden [Heij02] and my interviewees, the technical feasibility is taken for granted and is more or les an implicit factor (or hygiene factor). Furthermore the technical feasibility is mentioned as a product of universality and the source of "ease of use" factors.

Never the less this factor is critical to the spreading of the system to obtain a wide acceptance and a critical mass.

The integration effort, scalability and remote access are mentioned as primarily backend issues where interoperability relates to the actual device (the end-user front-end) and the end-users.

The end-user front-end must be interoperable with the greater part of all *mobile devices*, and the protocols and bearers of data transmission must be interoperable with the greater parts of the *different networks* (i.e. Bluetooth, IEEE802.11, GSM, 2.5G, 3G)
This leads us to two concise CSF:

- Interoperability of the front-end software and mobile devices (e.g. standards)
- Independency of network bearer and protocols used by front-end software (and back-end)

Finally the last factor is *performance*. The users are very critical in relation to user-friendliness and a slow performance is indeed critical for the user experience. The use of technologies (i.e. cryptology), the hardware and the way the application is designed is influencing the performance and the flow.

Example of Requirements

If the system has to be widely accepted, it must be compatible with *all devices* i.e. all the different brands. A MPS must be completely independent of the phone, to ensure highest possible distribution. The same argument is applicable in relation to the *networks*. The system must also be *independent of network* and *bearers* to ensure larger possible universality.

3.5 Universality

Universality is the result of the technical feasibility but the general scheme relates to the distribution of the system and the standards used for clearing/settlement issue (see the discussion of open vs. closed systems).

A universal system is a system that the major part of the mobile end-users is using and the major parts of the merchants are offering. A universal system also provides different kinds of payments. This implies that the end-users can use it across borders independent of the FSPs (banks, mobile operators or 3rd party e-cash systems). This issue leads to a concrete requirement about growing financial interdependence between mobile operators and/or banks [MaKr01].

If the payment system relies on the mobile operator as the FSP, the system shall be able to handle transaction from the same end-user (subscriber) on different competing networks. This requires the ability to roam the revenue [MaKr01].

If the payment system relies on an international credit card payment (e.g. visa), the cross border discussion is more or les irrelevant and the system can be considered as universal.
The system then builds on standards used by FSP - standards that enhance the interoperability. But an ideal situation, mentioned by most of the interviewees, was that the system should be build independently of any FSP.

Examples of Requirements

The system must support *micro and macro* payment. The end-user must be able to pay for small electronically services like news etc. or pay for more expensive goods i.e. tickets. It is more or less unimportant for the end-user if the FSP is a mobile operator or bank, but it is important that the FSP can settle payment from *all* end-users - i.e. *all banks or all mobile operators must support settling in the system*. Further the bank, mobile operator or a payment gateway (e.g. PSP) must support foreign end-users.

3.6 Summary

The table below summarizes the CSFs and the set of corresponding requirements:

CSFs	Examples of end-user requirements	Examples of merchant requirements	Examples of provider requirements
Ease of use	-Minimal initial set-up procedure -No installing -Intuitive -Few clicks	-As easy as existing internet payment systems	
Expenses	-No set-up fees -No transaction fees	-Minor set up fee -No extra transaction fees or running fees	Reasonable development/ maintenance costs
Security	-End-to-end encryption and/or -User authentication (-Verification / Non repudiation)		
Technical feasibility	-Compatible with all devices -Bearer and network independent -Performance (Quick data transmission)		
Universality	-Compatible with all mobile operators or -Compatible with all banks or -Compatible with international credit card clearing/settlement -Micro and Macro payment		

Table 10: CSFs and their corresponding requirements for m-payment systems.

4 J2ME™ and the Critical Success Factors

The points and conclusions achieved so far will now be used to analyse the suitability of J2ME™ as the client technology in an MPS.

This section will first list the basic potentials of J2ME™, to illustrate what makes the technology interesting for m-payments. This is followed by an analysis of J2ME™ in relation to the different CSFs and requirements.

The target platform for J2ME™ in this section is the CLDC[14] platform. The CLDC is the configuration layer that defines the minimum set of Java virtual machine features and Java class libraries available on a particular category of devices [SuCLDC00]. When referring to a J2ME™ application (hereafter MIDlet), it is meant primarily being built from the official MIDP1.0 API but also from available OEM specific APIs, which are compatible with the CLDC.

4.1 Why J2ME™ and M-payments?

Why consider J2ME™ in the first place, when we already have WAP, SMS or SAT technologies to handle m-payments? The foundation and ideology of Java and J2ME™ brings itself a reasonable set of *potentials* of being a part in a MPS.

The Java programming language has existed since May 23, 1995[wwwsu], but the official final CLDC 1.0a and MIDP1.0a APIs was released in May and September 2000 [wwwsu2]. According to all my interviewees, the press and general signals from the market, J2ME™ has a massive focus and is considered as an attractive technology for building mobile applications.

[14] CLDC (Connected Limited Device Configuration) is for the smallest wireless devices with 160 KB or more memory and 16/32 bit processors (i.e. mobile phones and PDA).

But there are several concrete arguments that indicate why J2ME™ (at all) should be considered as a interesting supplement for m-payments - these are [SuCLDC00] [SuDesig02] [SuMIDP1]:

- *Lower network usage and server load*: J2ME™ based applications can operate when disconnected and only interacts with a server when necessary. J2ME™ has its own runtime environment and the possibility of storing data in the mobile device.
- *Enhanced user experience*: The J2ME™ API provides enhanced possibilities for presenting GUI's like event handling and richer graphics [Sun02] [SuMIDP1].
- *Universality*: The details of machine architecture, operating system, and display environment are all handled transparently by the Java virtual machine (JVM). The same MIDP m-payment client can run on all MIDP-compliant devices - for example the same application can run on: different types of cell phones, two-way pagers or palmtops [SuDesig02] [SuMIDP1]. This allows m-payment system providers to target a wider range of end-users.
- *Internet*: Java is designed with a high focus on networking e.g. via HTTP or HTTPS, and Java's multi-platform capability makes it a natural choice for applications transferring data to use on the WWW.
- *Persistent storage*: The official MIDP1.0 API provides facilities for persistent storage (record store) of data [SuMIDP1]. The integrity of the record stores is kept throughout the normal use of the platform, including reboots, battery changes, etc. and is independent of any SIM/WIM [SuMIDP1].

There is a whole industry devoted to porting Java to new platforms, improving Java compilers and run-time systems, writing libraries of Java objects and fixing bugs.

But these above-mentioned arguments are not quite sufficiently enough to determine the suitability of J2ME™ in relation to m-payments - they only justify *considering* J2ME™ as an m-payment building block.

4.2 Does J2ME™ fulfil the Requirements for M-payments?

Each CSF will now be treated in the following, to see how J2ME™ can comply with the according requirements.

4.2.1 Ease of use of MIDlets

There are two relevant aspects in relation to "ease of use", when considering the use of MIDlets. These are downloading/installing, and the overall usability to carry out the transaction.

A MIDlet is downloaded and installed in a single procedure. A MIDlet is downloaded by requesting a URL directly to the specific MIDlet file (i.e. the .jad file) - For example: HTTP://www.java4mobile.com/paymidlet.jad file). The URL can be fetched in any WAP-browser, but a manual key in of the URL may not be optimal. According to the requirements a buy may only last a few clicks. Entering a URL on a mobile phone can be noticeably inconvenient and may not be well accepted by the end-users.

When using the push-possibilities in WAP 2.0, a URL can be pushed directly to the phone, and the user only has to click once to download the MIDlet.
Some WAP-gateways may not support MIDlet downloads why it also may be necessary to push the settings for a new WAP-gateway.
Another solution could be to push a URL via SMS, but this is a matter of native SMS- and WAP-functionality and thereby dependant of the manufacturers specific implementation[15].

But one thing is user involvement to retrieve the MIDlet - another thing is user involvement to initiate the concrete payment. The more possible data initially pushed to the phone, the easier it gets for the end-user. To enter a URL may not be acceptable, but entering a phone number for SMS may be acceptable - at least

[15] Motorola J2ME™ phones do officially support push of URLs via SMS.

according to the success of prsms services. Or even just tell the phone number to the merchant verbally at POS or via the internet. A fast and feasible way to push content and/or initiate a buy is via SMS, which is specified in the JSR120 [JcpJsr120]. Those of the interviewees who represented mobile manufactures indicated clearly that their phones would support the JCP120 in the near future.

A MIDlet it is more explicit in what kind of input is required - i.e. in a MIDlet you do not need to press a key multiple times to get a specific digit. For example "TextField.ANY "signals that any character can be entered, and "TextField.NUMERIC" signals that only digits can be entered[16]. This is useful as when a text field e.g. requires a phone number, then the phone keys only keys in digits and therefore input can be entered faster.

Compared to WAP and SAT, the MIDP 1.0 API provides enhanced GUI and UI possibilities. Like in Java applets, MIDlets provides dynamic event handling and possibilities for graphics and dynamic image drawings, which may enhance the usability and user experience.

It is essential, that the application flow is easy, intuitive and quickly to carry through. The High-Level UI API provides the standard and most common UI's for user interaction like: forms, text boxes, alerts, implicit/multiple choice lists, interactive gauge etc[17].
If any graphics are used, they are generated locally on the device, why network bandwidth usage is reduced and the performance is enhanced.

The rich alternatives of UIs and graphics provide the foundation of complying with a wider range of use cases and possible end-user demands. The storing facilities

[16] TextField is a Class in Javax.microedition.lcdui from the MIDP 1.0 [SuMIDP00].

[17] Classes from the javax.microedition.lcdui, MIDP 1.0 [SuMIDP00].

provide also the possibility of storing a concrete buying-session, prepared for eventual later resumption of shopping and payment.
In comparison to WAP, there's only a very limited dynamic storing facility. Some WAP-browsers provides only a few cookies for storage, but with a chance that they can be overridden and not all mobile phones will support them [wwwde].

4.2.2 J2ME™ and Expenses

As explained in the section of CSFs, the expenses depend first of all of the actual business case.
Apart from eventual expenditures imposed from the system providers, the only extra *direct* cost for the end-users is related to the airtime when *downloading the MIDlet* and *connecting* to the PSP/FSP when initiating or carrying out the payment. Besides connection fees and transfer speed, the download depends of the size of MIDlet and number and types of connections depends on the actual design of the MIDlet.

A payment MIDlet does in theory not have to be expensive in relation to connection and download. I have developed a prototype of a payment MIDlet, for the purpose of reaching a further knowledge of the development process and required MIDlet size. The prototype is developed by following the API specifications and interface description of a specific payment gateway provided by a Danish internet payment provider (PSP) [wwwdi].

A screenshot of the payment MIDlet:

Figure 5: Screenshot of the prototype of a payment MIDlet. See "Appendix F" for the source code.

The payment MIDlet provides interface and forms to collect credit card informations. The credit cards informations inclusive merchant data and payment data is posted via HTTPS connection to the PSP. The MIDlet is developed with the most basic functionalities to act as a *payment terminal*. The MIDlet does not function live at the present, but follows the MIDP 2.0 API draft7 for enabling HTTPS connections [JcpJsr118]. The MIDlet does not consider the payment initiation either. The corresponding "real" MIDlet may be larger, depending on the implementation of payment initiation.
The only conclusion available at the present from the prototype is, that is size is relative small (2.76kb) and is downloaded in less than a minute via GSM. The MIDlet carries out the transaction a single request and the corresponding response shows the transaction results in the screen [see Appendix F for further documentation].

Compared to WAP-based payment, all business logic is fetched from the web-server and usually no new software or hardware is required on the device. Because WAP fetches all business logic from the server, it may require more airtime compared to MIDlets, why this in theory could be more expensive.

New hardware may be required for WAP-payment, if the application logic depends on a wallet or keys stored in a SIM/WIM. The end-user could then be compelled to upgrade the SIM.
No extra costs are related to install/download a SAT-based application. SAT-based applications are usually using SMS as data bearer why "air time" when transferring data to PSP/FSP, is the price for each SMS - usually prsms[18].

For expenses in relation to development see "J2ME™ and Technical Feasibility" below.

4.2.3 Secure M-payment with J2ME™

As mentioned in the section of requirements, a MPS shall provide *end-to-end encryption* if any confidential data has to be transmitted. *User authentication* facilities or *verification/non-repudiation* is required if the mobile device is used as a wallet or PTD.

Cryptology

The MIDP 1.0 API does not provide officially classes or packages for cryptology. There are two relevant JSRs (Java Specification Requests) in relation to cryptology and secure m-payments according to the Java Community process(SM) Program:
- Mobile Information Device Profile, MIDP 2.0 (JSR 118) [JcpJsr118]
- Security and Trust Services API for J2ME™ (JSR 177) [JcpJsr177]

The JSR 177 is focusing at developing an API for secure execution, such as cryptographic operations to support payment protocols, data integrity and data confidentiality [JcpJsr177]. This is expected to be a part in the MIDP 2.0 [JcpJsr118] and will for example provide official implementation of HTTPS.

[18] The price for an mPay transaction is DKK 1,- + the standard rate for the SMS.

But the fact that no official cryptology API for the MIDP 1.0 has been released from Sun Microsystems, does not mean that MIDlet-applications cannot implement cryptology at present.
Given the importance of HTTPS in relation to m-commerce, Sun Microsystems have added an unofficial and support for HTTPS (kssl) as a part of the MIDP 1.0.3 reference implementation and the J2ME™ Wireless Toolkit version 1.0.3+ [Ma02]. HTTPS is not required by the MIDP 1.0 specification but if device manufactures releases devices supporting HTTPS, they will in theory be able to carry out secure transactions[19].

Support for HTTPS connections is required in the MIDP 2.0 API [JcpJsr118]. HTTPS must be implemented by one of the following specifications: HTTP over TLS, SSL V3, WTLS or TLS over Profile and Tunnelling Specification [JcpJsr118].

It is important to be aware of that the transmission is routed through a WAP-gateway before it connects to a web-server processing the payment transaction if end-to-end encryption has to be obtained via WTLS (and WAP). A 100% end-to-end WTLS transaction between a phone and a payment-server via the internet is difficult, due to the fact that the WAP-gateway has to establish another secure connection - in this situation the risk for *eavesdropping* occurs. And as shown above in the concrete MPS examples, a SAT-transmission must likewise be routed through a SMSC before it connects to the web server.
WTLS and SSL is subdued to the US export restrictions for cryptographic software, where symmetric keys are limited to 40 bits. But due to the restrictions the most used encryption ciphers in WTLS are RC5_CBC with 40- and 56-bit keys and DES_CBC with 40-bit key [Ri01]. Usually ciphers in CBC-mode are vulnerable and DES has been proven cracked by the Electronic Frontier Foundation (EFT) [Ri01].

[19] According to an unofficial compatibility test [wwwki],the following devices supports HTTPS: Motorola i50sx / i55sr / i85s /i95cl /V60 / V66 and T280.

HTTP over SSL is the most widespread secure technology for payment transactions via the internet. SSL provides *confidentiality* (end-to-end encryption), *integrity,* server certificate and the option for client certificate [NeISSL].

One of the above-mentioned advantages of Java is that whole industries are focusing on improving and increasing different APIs. A concrete initiative called Bouncy Castle has released a lightweight API (BC-API) with cryptology and certificate facilities, designed for J2ME™. The BC-API is widely accepted and articles describing the BC-API are e.g. published at the official J2ME™ website of Sun [Ge01], at the IBM developerWorks [YuLo02] and in the book "Wireless Java" by Jonathan B. Knudsen (chapter 12) [Kj01].

The BC-API provides a security toolbox obtained from the original Java Cryptography Architecture (JCA) and the JAVA Cryptography Extension (JCE) and has been boiled down to support the CDC and CLDC devices [BcSpec02]. The following tools are supported [BcApi02]:

- The most accepted chiper engines for cryptology e.g. DES, Triple DES, RSA, RC4 and RC6 (see "Appendix A" for complete list of ciphers)
- The most accepted digest engines for generating HASH values e.g. MD5 and SHA-1 (e.g. for digital signatures) (see "Appendix A" for complete list of digest)
- Support for Message Authentication Codes (MAC) generation.
- Support for key generation and exchanging e.g. RSA or Diffie-Helman keys
- Support for reading and writing certificate e.g. x509, X.9.62 and PKCS#12

The BC-API is compatible with the CLDC/MIDP1.0 and not dependant of manufacturer specific implementations.

Encryption relies of a corresponding key (symmetric or private key) that is accessible to the MIDlet during runtime. J2ME™ provides facilities to use and store encryption

keys. Keys can be stored and updated in the record store as mentioned above, or they can be stored in unique resource file generated at deployment[20].

Authentication

Authentication refers to the process when the end-user identifies him/herself towards the mobile phone and "opens" e.g. the wallet or payment MIDlet. Different techniques can be used for Authentication but a PIN is mostly used in standard payment systems and is a natural option for m-payment authentication as well.

As mentioned above, the MIDP 1.0 API provides facilities for persistent storing in a record store. The record store is a single file associated to the corresponding MIDlet or MIDlet suite. When a MIDlet suite is removed from a device all the record stores associated with its MIDlets will also be removed. The MIDP 1.0 API only allow the manipulation of the MIDlet suite's own record stores, and does not provide any mechanism for record sharing between MIDlets in different MIDlet suites. MIDlets within a MIDlet suite can access each other's record stores directly [SuMIDP1].

In contrast to SAT or WAP applications, the record store is independent of any SIM or WIM, why PIN scenarios may be more flexible. The MIDlet vendor has therefore a lager range of use-case scenarios in relation to the PIN - e.g.:

- Dynamically updating of PIN, E.G: via user from internet (OTA) or via phone, from PSP via OTA etc.
- Multiple PINs and Multiple users - Dynamically adding or deleting of users.

SAT or WAP uses SIM/WIM to store data for authentication [WapWIM01] [Gempl01]. To use SIM/WIM depends on further APIs and individual implementation from the manufacturers. A downside of this may be, that a MIDlet capable of SIM storing may not be compatible with current J2ME™ phones.

[20] But storing of sensitive data in the record store brings along some security threats - see the "Security risk with J2ME™"-section below.

Verification / Non-repudiation

Verification constitutes the foundation for non-repudiation and requires storage facilities reachable from the MIDlet. The storage shall e.g. keep a private key to digitally sign the verified data. A payment-MIDlet is therefore able to support non-reputable verification also by using a resource file or the record store.

The BC-API provides tools to generate non-reputable data. The most obvious tools in the BC-API are SHA#-Digest or MAC values (see "Appendix A").

4.2.4 Security Risk with J2ME™

Unfortunately there's a considerable security risk in using the MIDlet itself to store confidential data - e.g. by distributing the MIDlet containing a hard-coded private key or PIN. It is theoretically possible to download a MIDlet (or MIDlet suite) to a PC from the mobile phone via standard data cables. Even though the MIDlet is compiled and obfuscated[21] a serious hacker could theoretically restore any data by reverse engineering and copy the PIN or private key.

A 100% secure distribution of MIDlets is not possible either. The only secure distribution is via WTLS at the present, if a MIDlet shall be distributed containing secret data (e.g. PIN or private key). As explained above WTLS does not provide end-to-end security to web servers and the encryption ciphers are demonstrably broken.

[21] Obfuscation is for preventing reverse engineering. An obfuscation process strips all unnecessary information (line numbers, local variable names and source file names used by debuggers) from the classes. Also, class, interface, field and method identifiers are renamed to make them meaningless [Re02].

4.2.5 J2ME™ and Technical Feasibility

Interoperability - Devices and Network

Like the standard Java technologies, J2SE™ and J2EE™, the J2ME™ platform provides the advantages of build-in consistency across platforms. MIDP1.0 applications run the same way on any platform that is based on CLDC/MIDP1.0. All MIDP implementations must pass all tests from the Technology Compatibility Kit provided by sun [MP02].

The fact that MIDlets can run in any J2ME™ compatible device enlarges the potential target audience and opens a whole new dimension in relation to m-payments. M-payments with J2ME™ are not restricted to be carried out by mobile phones (See "Appendix B" for a list of CLDC and MIDP 1.0 devices).

But in spite the intension of platform independency, some hardware dependent features in the present mobile phones is indisputable different and cannot be covered in the standard CLDC/MIDP specifications. The results must be tailored MIDlets to act seamless in the different phones. Concrete examples of this are e.g. the different screen sizes (see "Appendix B") and low-level interaction with different keys or other hardware specific functionalities.
This issue may affect the development cost, but probably mainly in relation to games and entertainment MIDlets - Payment MIDlets may, other things being equal, depend less on the screen size and special keys. If developers are following the standard APIs, it may in theory be relatively uncomplicated to develop and test J2ME™/MIDP applications. All SDKs and APIs are free of charge and no extraordinary cost is imposed. The same conditions are valid for WAP developing as well.
The SAT Developer Kit requires a complete simulated GSM environment, including GSM network, SMSC and a mobile phone [gempl01]. Further SAT application are reserved to mobile operators.

SAT applications are stored in the SIM and therefore only targeting mobile phones [gempl01].

In relation to network connections all MIDP1.0 devices are providing support for HTTP 1.1 connections, which is a standard requirement defined in the MIDP1.0 specification [SuMIDP1] (see "Appendix B" for a list of MIDP devices).
But J2ME™ is designed independent of the network carriers. J2ME™ is "indifferent" about what bearer is being used to transfer data and can use different kind of protocol stacks and hardware to connect to external networks.

An official SMS API (Wireless Messaging API) for sending and receiving SMS via MIDlets has just been released from the JCP [JcpJsr120]. But at the present there's no required support for JSR120 in the MIDP 1.0 devices. It depends on the manufacturers implementation of the JSR120 if MIDlets shall use SMS as data bearer.

An official MIDP Bluetooth API [JcpJsr82] has also recently been released[22]. But like the SMS API, there's no required support for the JSR82 [JcpJsr82]. The use of bluetooth depends also of the manufacturers implementation of the API specification. It depends both on new API specification from JCP and the implementation from the manufacturers if J2ME™ has to use other network standards like e.g. infared (Irda) or IEEE802.11 (Wireless LAN).

In contrast SAT is dependent on SMS as a bearer and therefore also restricted to the limit of 160 characters. SAT applications are therefore written and optimised/limited for SMS.

[22] Supported protocols: L2CAP (connection-oriented only), RFCOMM, SDP, Object Exchange protocol (OBEX). Profiles supported: Generic Access Profile (GAP), Service Discovery Application Profile (SDAP), Serial Port Profile (SPP) and Generic Object Exchange Profile (GOEP) [jcpJsr82].

J2ME™ is not a browser, but an independent application. J2ME™ has though the ability to parse any kind of content - as long as the containing data is well defined. J2ME™ can parse the content from e.g. HTML-, WML- or XML-documents.
At the present several XML-parser APIs has been provided to MIDP 1.0 (see "Appendix C"). And according to proposed Specification for the MIDP 2.0, an XML-parser will be included in the MIDP 2.0 API [JcpJsr118].

Scalability and Performance

A MIDlet can run in standalone mode, which results in fewer users accessing servers at PSP at any given time. This in turn improves performance and scalability for the payment server, and reduces demand for network bandwidth.
MIDlet applications are also upwardly scalable to applications build in the Java 2 Standard Edition (J2SE) or Java 2 Enterprise Edition (J2EE) – of course with rework. MIDlets are independent of e.g. SIM card or other hardware at runtime. An upgrade of a SAT application could by unfortunate circumstances result in the need of new SIM card.

WAP is in contrast dependent of server connection, and could suffer from possible network instability. WAP may therefore be a bit slower but provides the ability to present cashed WAP-sites.

Possible network instability could provide errors during the transmissions. Especially m-payment applications must be geared up to handle these problems. Wireless network protocols may be able to detect and correct some errors, but the error handling in Java coding is capable of addressing all the kinds of transmission errors and presenting them to the end-user.

Further J2ME™ is a multithreading environment. This allows a MIDlet to progress user interaction while other important processes like e.g. network access are running in the background.

Heavy computations for e.g. encryption purposes carried out by MIDlets can result in slower response rates [GuGu01].
I have carried out a concrete "speed test", with the purpose of measuring the actual performance of a MIDlet. I have selected four of the most common ciphers[23] and their suitable key lengths from the BC-API. The speed for encryption computations is measured. The encrypted text is a 23-digit number to simulate a standard credit card transaction: 16 digits for card-number, 4 digits for expiring date and 3 digits for a control-number. The test is carried out in a MIPD 1.0 environment with a Nokia 6310i.

The result are listed in the table below:

	64 bit key	**128 bit key**	**256 bit key**
DESengine (15kb)	~405 millisec.	-	-
RC4engine (6kb)	~113 millisec.	~116 millisec.	-
RC6engine (12kb)	~299 millisec.	~301 millisec.	~304 millisec.
DesedeEngine(16kb)*	-	~468 millisec.	-

Table 11: The time elapsed during encryption in a MIDlet.
See "Appendix D" for the source code and further documentation.

*** Triple DES.**

The above results from the test indicate apparently, that the time used for credit card encrypting with symmetric keys is not perceptible.

4.2.6 J2ME™ and Universality

The interesting question in this case is: does J2ME™ affect whether the system will be a "worldwide" payment system, with the support of *open payment systems*

[23] The RC4, DES and Triple DES are common during SSL connections [wwwrs]

(hereunder multiple FSP, payment across borders) and *micro*- and *macro* m-payment?

First of all, an important point about the choice of technology is whether the FSPs or PSPs will accept the security level of the system. As mentioned above, several techniques to ensure security are yet possible and compared to existing internet payments, provides at least same level of security.

As mentioned in the section of *interoperability – devices and network*, J2ME™ targets a broader range of end-users via enhanced compatibility of networks and devices. These circumstances also affect the possible targeting of a *wider range of PSPs and FSP*s. For example, merchants can process payments by using any existing internet payment gateway/PSP[24] with standard security certificates, by building a "payment MIDlet" based on the *terminal* function using HTTPS. The same circumstances are valid for whether the payment is a *micro or macro payment* and the product is *physical or electronically*. Electronically or interactive products are generally hosted at the web, why payments is done most feasible via the web.

In contrary, WAP or SAT m-payments require that PSPs provide either WAP-gateway or SMSC functionality[25]. This issue conflicts the overall purpose of "*universality*" by narrowing down the possible amount of PSPs.

Secure SAT solutions depend on storing e.g. symmetric keys in the SIM card[26]. The fact that SIM cards are provided by the belonging mobile operator, brings along two problems about *universality*:

[24] As mentioned a MIDlet is not a browser, but supports http POST/GET. A payment via the MIDlet is possible by using e.g. XML pages and an XML-parser instead of html pages to present user information.

[25] To ensure real end-to-end encryption from the mobile device to the PSP.

The target user group are restricted to one group of subscribers, and the mobile operator is not an official trusted CA provider.
Mobile operators cannot issue trusted certificates with a private key, or if the system is based on symmetric keys, only the single mobile operator knows the belonging key.

As mentioned several times above during the analysis of J2ME™, all phones do not support the same APIs or functionalities yet.
The CLDC guarantees the portability and interoperability of MIDP APIs across devices [SuCLDC00]. All J2ME™ phones don't necessarily support additional released MIDP compliant APIs or features, which isn't supported by the present CLDC 1.0 or required in the MIDP1.0.
Those of the above-mentioned MIDP APIs that specifies the support for HTTPS, SMS, push, or bluetooth, are not required features in the current CLDC/MIDP1.0 phones. If these features are included in a MPS today, the system will not be interoperable with all J2ME™ phones - but hopefully in the near future.

[26] With the fact in mind, that IBM has claimed that it is possible to crack a secret PIN or key in the SIM card [wwwib].

4.3 Summary

The table below shows a review of the most important strengths and weaknesses in J2ME™ in relation to the CSFs.

	J2ME™ - CLDC/MIDP 1.0	
	Strengths	*Weakness*
Ease of use	-Simple user installation -Rich, enhanced GUI -Easy input	-No push deployment (yet) - SMS or push not required in MIDP 1.0
Expenses	-Inexpensive for end-users -Inexpensive for developers	
Security	-Strong encryption -End-to-end (kssl) -Authentication -Verification	-Insecure persistent storage -Insecure deployment -HTTPS not required in MIDP 1.0
Technical Feasibility	-Device independency -Network/bearer independency -Persistent storage -Easy development -Multithreaded -Displaying web content	- Device dependency in relation to screen size and low-level interaction.
Universality	-Targets a larger range of payment gateways (PSP)	

Table 12: Strengths and weaknesses in CLDC/MIDP 1.0

The next table below show a superior comparison of CLDC/MIDP 1.0, WAP and SAT in relation to the CSFs, with the weaknesses of J2ME™/MIDP 1.0 in mind. The comparison is illustrated by a small point-scale from 1-3. Each point is illustrated with a "+", where 3 points ("+++") means "best" or "most feasible".

	J2ME™	WAP	SAT
Ease of use			
-Deployment	+	+++	+++
-User interface	+++	++*1	++
-Payment initiation	+++	++	+++
Cost			
-Users	+++	++	++
-Developers	+++	+++	+
Security			
-Encryption strength	+++	++	+++
-End-to-end	+++	++	++
-Authentication	+++	(++)*2	+++
-Verification	+++	(++)*2	+++
Technical Feasibility			
-Device independency	++(+)	++	+
-Network/bearer independency	++	+	
-Storing facilities	+++	+	+
-Performance	++	+	+++
-Easy development	+++	+++	+
-Multiple web content	+++		
-Multithreaded	+++		
Universality			
-Possible collection of PSP/FSPs	+++	+	+

Table 13: J2ME™ / WAP / SAT comparison

***1) By general adoption of WAP 2.0 further UI enhancements for WAP becomes possible.**

***2) Authentication and verification via WAP depends on storing facilities in SIM or WIM.**

5 Conclusion

In the following the conclusion is succeeded by an outlook, to finish with a larger perspective beyond this research scope. The two elements are covered in separate sections below.

5.1 Conclusion

Mobile payment systems are not isolated systems, but systems with a strong interdependence of existing internet payment systems. Mobile payments are an extra option to existing payment methods and the mobile phone can be considered as another tool for carrying out payments.

A lot of requirements need to be fulfilled if the new m-payment system shall be able to compete with the existing payment systems. This thesis has focused on the requirements for the client technology in the mobile phone.

When using the mobile pone as a payment tool, the phone can act as a PTD or a terminal. In these situations the data handling and transmitting meets a lot of challenges. Many different initiatives and forums have been established to cope with these challenges, with the main focus of developing and distribute standards for m-payments - standards, which all the interviewees also agreed on, has to live up to a set of critical success factors, to be a successful m-payment system.
A bigger change for network externalities is present if all requirements from the critical success factors are met and accepted by the actors.

But m-payments are still in its infancy. As shown in the examples m-payment solutions are still being developed with standards defined of individual business segments. Therefore they often approach less significant markets – even though the mobile marketplace is global. The result is market fragmentation and is probably one of the reasons why m-payments still have been slow.

The J2ME™ technology has been chosen in this thesis as a possible m-payment technology, to see if J2ME™ could comply with the requirements from the market - and thereby be a part of a successful m-payment system.

Concrete J2ME™ applications are tested, evaluated and several strengths and weakness in relation to the set of market requirements are described.

There are two issues when answering the problem formulation in the light of the J2ME™ analysis - first the missing facilities of the present APIs, and second the facilities that is not implemented in the mobile phones.

Today, it is possible to implement and provide an end-to-end secure payment MIDlet, but it will not comply with the requirements of "easy to use" and "universality".

There are no acceptable deployment facilities for a payment MIDlet (with MIDP 1.0 and WAP 1.0) at the present, because this action requires the user to enter a URL to fetch the MIDlet. It will not be possible to push MIDlet URLs to the end-users and provide easy receiving and installing of MIDlets in a few clicks, until general adoption of WAP 2.0 or specific SMS/WAP push facilities are generally implemented.

Second easy initiation of m-payments is not generally available either. The JSR120 (Wireless API), which supports SMS, is available but not generally implemented in all phones. Until the release of MIDP2.0 or general implementation of the JSR120 has taken place, the initiation of a payment is not close enough to be "as easy as existing systems".

MIDlets provides rich GUI facilities and easy data entering, which may enhance the overall user experience. Further the Multi thread environment or storage facilities provide a wider range of use cases - online or offline.

J2ME™ does not carry along noticeable extra expenses for the end-users - for one thing because payment MIDlets in theory are small and require limited airtime. Developing J2ME™ payment applications without different hardware specific features may be relatively uncomplicated and inexpensive for developers. All SDKs are free of charges and is well documented with at large hooked developers community.

J2ME™ supports end-to-end encryption, authentication and verification. End-to-end can be SSL (kssl) or proprietary implementation from BC-API. Unfortunately HTTPS is not required in the MIDP 1.0 implementation why HTTPS at the present is depending on the manufacturers own implementation. HTTPS is required in the MIDP 2.0 API, which expects to be released ultimo 2002. The best possible implementation of HTTPS should be coordinated between manufacturers to ensure homogeneity across devices and compatibility of all secure J2ME™ applications. Further the implementation should avoid the WTLS specification to ensure end-to-end and independency of WAP-gateways.
The BC-API is compatible with all CLDC/MIDP1.0 platforms.

If using the phones as terminals with HTTPS, the security level is acceptable, but does not support the functionality and specifications of a Personal Trusted Device. Two things must be done to turn the mobile phone into a Personal Trusted Device with J2ME™ with the support for authentication, verification and non-repudiation: first the manufactures must secure the record store facilities by e.g. preventing copying of the MIDlet to a pc, and second, a secure download of MIDlet to the phone must be available.

An acceptable secure authentication or verification may be to use SIM/WIM. But unfortunately this is not suitable in relation to universality and there's no official API or implementation to provide connection between MIDlets and SIM/WIM at the present. A solution in the long run could be to introduce dual slot phones, with the support of reading smart cards containing secret keys or client certificates to ensure secure authentication or verification. Dual slot prevents the dependency of SIM and thereby the mobile operators, but causes the inconvenience to distribute a new generation of mobile phones.

Payment MIDlets may have a broader possible rage of users, because they can run in every device that supports the CLDC/MIDP1.0 specifications. If the MIDlets follows the official APIs, they run the same way in each device. This ensures run-time consistency across platforms and makes the development process less expensive and easier for the developers.

J2ME™ is bearer and network independent. J2ME™ provides APIs for SMS and Bluetooth in addition to the required support for HTTP connections. But this is a question of the manufacturers willingness to implement these functionalities in the devices.
J2ME™ is also Build for networking - It provides direct web connections and the possibility of fetching of content from e.g. XML, HTML etc. J2ME™ makes it easier to use existing payment infrastructure, by adapting internet APIs to existing PSPs and FSPs – J2ME™ is therefore not limited to either micro- or macro m-payments and mobile operators or banks.

The independency of bearer and network offers richer and more flexible payment MIDlets - different HTTP connections and the support for SMS, provides an opportunity to offer the end-user multiple payment scenarios, e.g. either micro/macro payment or premium rate SMS from the same payment MIDlet.

In relation to J2ME™ one of the key actors in the value chain for m-payment systems is the mobile phone manufacturers. The presently available J2ME™ APIs meets actually all the requirements - we just need the phones to meet some of the requirements too. The manufacturer decides how and when to implement and upgrade the implementation of the above-mentioned relevant J2ME™ APIs.

Concluding, the limitations and weaknesses are actually not in the J2ME™ but in the willingness of the manufacturers to implement the APIs and straightforward deployment facilities. Acceptable m-payment solutions with J2ME™ will be possible, if consistent support for HTTPS in MIDP 2.0 will be implemented.

5.2 Outlook

The last step of this thesis is to position the scope and content of this thesis in another perspective and to identify possible research areas in the future. The headlines will not cover all possible directions but try to grasp some of the most important perspectives.

Standardization

As mentioned in the thesis, the overall success of m-payments (or payments in general) is generally measured by the *number* of end-users using the payment system. The number of phones supporting the actual payment system should not be reduced by incompatible technologies.

A lot of forums have been constituted with the focus on the same case - which among others is to develop standards for m-payments. But these forums may likely develop different standards for the same purpose - and then we're probably back to scratch.

Although J2ME™ is platform independent the J2ME™ standard specifications cannot take the interaction with different hardware specifications into consideration. The question is how this can this be standardized? A further study may pay attention on

how to standardize or enhance the collaboration between the manufacturers, to fully utilize the platform independency of J2ME™.

The Business Case

A number of potential business cases for m-payments are present - e.g.: vendor machines, tickets, location based services and games/entertainment.
This scope of this thesis was to analyse J2ME™ for m-payment systems in general - the scope of a business case is to analyse a concrete business plan, describing the case, actors, business models and concrete system etc. The business case should further pay attention to who drive the introduction and pays the expenses of developing the system. The business case shall also describe how to introduce the system, educate and make the system visible for the users. Further interesting use-cases in relation to new deployment scenarios is interesting - e.g. dynamic deployment at POS via bluetooth.
The above issues are not complete, but the intension is, to make sure that the "concrete" payment business case is going to be a success.

The above examples are just two of the research topics that I think need to be addressed in the future. Surely many more can be elaborated in future research and in relation to the popularisation of m-payments.

6 References

- [AaEnHeHi02] Aarino, Antti & Enkenberg, Aki & Heikkilä, Jukka and Hirvola, Sanna, "Adoption and Use of Mobile Services Empirical Evidence from a Finnish Survey", University of Jyväskylä, proceeding of the 35th Hawaii International Conference on System Sciences 2002.
- [AnDi02] Ancar, Bill and D´Incau, David, "Value-Added services in Mobile commerce: An Analytical Framework and Empirical Findings from a National End-user Survey", Institute for Advanced Management Systems Research (IAMSR), proceeding of the 35th Hawaii International Conference on System Sciences 2002
- [AvFi95] Avison, D, Fitzgerald, G. "Information systems Development: Methodologies, Techniques and Tools" 2.ed, McGraw-Hill Pub.
- [BcSpec02] Bouncy Castle, The Legion of The, "Specification", HTTP://www.bouncycastle.org, v. 1.1.4, 2002
- [BcAPI02], Bouncy Castle, The Legion of The "API", HTTP://www.bouncycastle.org, v.1.1.4, 2002
- [BrMc01] Bray, Chris & McKeown, Paul & Palfreyman, Nobert, John & Winegust, Fred and Oliver, David "Wireless Payments - Money out of Thin Air?" IBM wireless e-business, June 2001
- [CaEY01] Cap Gemini/Ernst & Young, "Internet Mobile for Business: So far from expectations?", whitepaper June 18, 2001.
- [Dahl01] Dahlström, Erik , "The common future of wallets and ATMs? Mobile phones!", ePSO newsletter vol.02, sep 2001
- [Delbe75] Delbecq, A.L., *et al*. "Group Techniques for Program Planning: a guide to nominal group and delphi processes". Scott, Foresmann and Company, 1975
- [DeWiTe02] Dennis, Alan & Wixom, Barbara Haley & Tegarden, David, "Systems Analysis and Design, An object-Oriented Approach with UML", John Wiley & Sons 2002.

- [EcmlNokia01] "ECML in Nokia Mobile Wallet", Version 1.0, www.forum.nokia.com, Sep 2001
- [EvWu00] Evans, Philip & Wurster, Thomas S.,"Blown to Bits", Harvard Business School Press 2000.
- [Fdim02] "Microbetaling – Design, Grensesnitt for innholdsleverandør", version 0.2 Udkast, FDIM 5/2 2002.
- [Futu01] Future Lab Barometret juni 2001, "Mobile business", HTTP:www.futurelab.dk, juni 2001
- [Gempl01] GEMPLUS (2001). "Boost Value Added Services With SIM Application Toolkit", Version 3, White Paper, www.gemplus.com, May 2001
- [Ga01] Gartner, Cavalieri Hilton "Mobile Business, from the Mobile Internet to the supranet", Rome 28-29 june 2001.
- [Ge01] Giguere, Eric (2001). "Data Encryption for J2ME™ Profiles", wireless.java.sun, Dec 2001.
- [GuCr02] Guthery, Scott B. & Cronin, Mary j, "MOBILE APPLICATION DEVELOPMENT WITH SMS AND THE SIM TOOLKIT", McGraw-Hill 2002
- [GuGu01] Gupta, Vipul and Gupta, Syumit, "KSSL: Experiments in Wireless Internet Security", SMLI TR-2001-103, Sun Microsystems, November 2001
- [GuVi02]Gupta, Vipul and Gupta, Sumit, "Securing the Wireless Internet", Sun Microsystems 13/2 2002
- [Heij02] Heijden, Hans van der, "Factors affecting the successful introduction of mobile payment systems", Vrije Universiteit Amsterdam, 2002
- [ITUecml99] ITU, HTTP://www.itu.int/newsarchive/wtd/1999/iht10/tfi-10.html, 1999
- [Ja00] Jacobs Kai, "Information Technology Standards and Standardization: A Global Perspective", Idea Group publishing 2000.
- [JcpJsr177] Java Community Process(SM) Program, "JSR 177: Security and Trust Services API for J2ME™", HTTP://www.jcp.org/jsr/detail/177.jsp, April 2002.

- [JcpJsr118] Java Community Process(SM) Program, "JSR 118: Mobile Information Device Profile 2.0", Draft 7, HTTP://www.jcp.org/jsr/detail/118.jsp, Feb 2002.
- [JcpJse120] Java Community Process(SM) Program, " Wireless Messaging API", HTTP://www.jcp.org/jsr/detail/120.jsp, Jun 2002.
- [JcpTCK] Java Community Process(SM) Program, " TCK Tools & Info", HTTP://www.jcp.org/resources/tdk/, 20002
- [JcpJsr82] Java Community Process(SM) Program. "Java TM APIs for Bluetooth TM Wireless Technology (JSR-82), Specification Version 1.0a, Java TM 2 Platform, Micro Edition,", April 5, 2002
- [Ka00] Kare-Silver, Michael de, "E-Shock 2000", Macmillan Press LTD 2000.
- [KeMa01] Keen, Peter G.W., Mackintosh, Ron, "The freedom Economy", Osborne/McGraw-Hill 2001.
- [Kn01] Knudsen, Jonathan B., "Wireless Java : Developing with Java 2, Micro Edition", chapter 12, APress, June 2001
- [Kn02] Knudsen, Jonathan B. “Parsing XML in J2ME™[tm] XML in a MIDP Environment”, HTTP://wireless.java.sun.com/midp/articles/, March 2002.
- [Ma02] Mahmoud, Qusay H. "Secure Java MIDP programming using HTTPS with MIDP", http://www.wireless.java.sun, june 2002
- [Ma01] Malte, Krueger, "The Future of M-payments – Business Option and Policy Issues" Background Paper No.2, Electronic payment Systems Observatory (ePSO) August 2001.
- [MaCa01] Malte, Krueger, Carat, Gérad, "M-Payments and the role of mobile operators", ePSO newsletter vol.02, sep 2001
- [MetPTD] MeT, Mobile electronic Transactions, "MeT PTD Definition", version 1.1, http://www.mobiletransaction.org, Feb. 2001
- [Mo02] Mortier, Peter, "WAP and J2ME™", HTTP://mexeforum.org/articles.htm", 2002.
- [NeHSSL] Netscape, "How SSL Works", HTTP://developer.netscape.com

- [NeIPKI] Netscape, "Introduction to Public-Key Cryptography", HTTP://developer.netscape.com
- [NeISSL] Netscape, "Introduction to SSL", HTTP://developer.netscape.com
- [NeSSL] "SSL 3.0 specification", Netscape 1996, HTTP://developer.netscape.com/tech/security/ssl/protocol.html
- [NoSMS] "Nokia SMS API, v0.9", Nokia Corporation, HTTP://www.forum.nokia.com
- [NoUI], "Nokia UI API, v0.9.2 I API", Nokia Corporation, HTTP://www.forum.nokia.com
- [NoWal] "Nokia phones with wallet", Nokia Corporation, www.nokia.com/mobilecommerce/wallet.html
- [PBS02] "PBS Secure Mobile Payments Solution", PBS, marts 2002
- [PbsPsip] "Merchant Guide for SSL Based Card Acquiring (PSIP)", Version 1.0, PBS
- [PaEa02] Parsons, Jon W. (AmEx) & Eastlake, Donald E. 3rd (Motorola), "Electronic Commerce Modeling Language (ECML)" Version 2, HTTP://www.ietf.org/internet-drafts/draft-ietf-trade-ecml2-req-05.txt, May 2002
- [Pay01] Paymentgroup, "Guide to the Payment Interface", Version 0.7, www.paycircle.org, November 2001
- [PeMeTh02] Pedersen, Per E. and Methlie, Leif B. and Thorbjørnsen, Helge "Understanding mobile commerce end-user adoption: a triangulation perspectiveand suggestions for an exploratory service evaluation framework", Agder University college and Foundation for Research in Economics and Business Administration, proceeding of the 35th Hawaii International Conference on System Sciences 2002
- [Prefe01] "The Preferred Payment Architecture Executive Summary, Requirements for manufacturers and standardisation bodies" version 1.0, Workgroup Executive Mobey forum June 2001

- [Re02] Retrologic , "RetroGuard User's Guide", HTTP://www.retrologic.com, 2000.
- [Ri01] Richter, G., "Evaluation And Implementation of Secure Mobile Commmerce Systems", Department of Electrical, Electronic and Computer Engineering, University of Pretoria, November 2001.
- [ShSw98] Shon T.W. and Swatman P.M.C., "Effectiveness Criteria for Internet Payment Systems", Internet Research: Electronic Networking Applications and Policy, Vol. 8, No. 3, 202-218, 1998.
- [ShVa99] Shapiro, Carl and Varian, Hal R., "Information Rules". Boston, MA, Harvard Business School Press, 1999.
- [SsWr98] Schmidt, Susanne K. and Werle, Raymund, "Coordinating Technology. Studies in the International Standardization of Telecommunications", MIT press 1998.
- [SuCLDC00] Sun Microsystems, Inc. "Connected, Limited Device Configuration (CLDC) Specification, version 1.0a", Sun Microsystems, Inc., may 19, 2000.
- [SuDesig02] Sun Microsystems, "Designing Wireless Enterprise Applications Using java™ Technology, HTTP://java.sun.com/blueprints/, Jan 2002.
- [SuMIDP1] Mobile Information Device Profile (MIDP) Specification ("Specification"), Version: 1.0, Release: September 15, 2000, Copyright 2000 Sun Microsystems, Inc.
- [TrSt98] Treese, G. Winfield & Stewart, Lawrence V.," Designing Systems for Internet Commerce", Addison Wesley 1998.
- [Umts11] UMTS Forum, "Enabling UMTS Third Generation Services and Applications", No.11, HTTP://www.umtsforum.org, October 2000
- [UnWi02] UnWire, "UnWire SMS gateway – Version 0.84", Interface description, UnWire 2002.
- [Wap2.0] Wireless Application protocol Forum Ltd (2002), "Wap 2.0 Technical White Paper", www.wapforum.org, jan 2002.

- [WapWIM01] Wireless Application protocol Forum Ltd, "Wireless Identity Module Specification, WAP-260-WIM-20010412-1", Version 12-july-2001.
- [WapWMLs01] WMLScript Crypto API Library Specification, WAP-161-WMLScriptCrypto-20010620-a, Version 20-Jun-2001.
- [WapWTLS02] Wireless Application protocol Forum Ltd, "Wireless Transport Layer Security Specification (WAP-261-WTLS-20010406-a)", Version 06, Apr-2002.
- [wwwbe] www.betaling.dk
- [wwwde] www.devx.com
- [wwwco] www.coinclick.dk (Content Billing A/S)
- [wwwdi] www.dibs.dk, Architrade A/S
- [wwwib] www.ibm.com/news
- [wwwie] The Internet Engineering Task Force, HTTP://www.ietf.org/
- [wwwIt] www.it-leksikon.dk (IT-dictionary)
- [wwwki] http://kissen.cs.uni-dortmund.de:8080/devicedb/index.html
- [wwwrs] www.rsasecurity.com
- [wwwSe] www.searchebusiness.com (business-dictionary)
- [wwwsu1] http://java.sun.com
- [www.su2] http://wireless.java.sun.com
- [wwwte] www.teleindustrien.dk
- [wwww3] www.w3.org
- [YuLo02] Yuan, Michael Juntao & Long, Ju, "Security challenges and solutions for mobile commerce applications", IBM deveoperWorks, HTTP://www-106.ibm.com/developerworks/wireless/library/wi-secJ2ME.html, June 2002.

Appendix A: Bouncy Castle project

Symmetric block chiper engines:

Name	KeySizes (in bits)	Block Size	Notes
AESEngine	0 .. 256	128 bit	
AESWrapEngine	0 .. 256	128 bit	Implements FIPS AES key wrapping
BlowfishEngine	0 .. 448	64 bit	
CAST5Engine	0 .. 128	64 bit	
CAST6Engine	0 .. 256	128 bit	
DESEngine	64	64 bit	
DESedeEngine	128, 192	64 bit	
DESedeWrapEngine	128, 192	64 bit	Implements Draft IETF DESede key wrapping
IDEAEngine	128	64 bit	
RC2Engine	0 .. 1024	64 bit	
RC532Engine	0 .. 128	64 bit	Uses a 32 bit word
RC564Engine	0 .. 128	128 bit	Uses a 64 bit word
RC6Engine	0 .. 256	128 bit	
RijndaelEngine	0 .. 256	128 bit, 160 bit, 192 bit, 224 bit, 256 bit	
SkipjackEngine	0 .. 128	64 bit	
TwofishEngine	128, 192, 256	128 bit	
SerpentEngine	128, 192, 256	128 bit	

Symmetric stream chiper engines:

Name	KeySizes (in bits)	Notes
RC4Engine	40 .. 2048	

Asymmetric block chiper engines:

Name	KeySizes (in bits)	Notes
RSAEngine	any multiple of 8 large enough for the encoding.	

Digest engines:

Name	Output (in bits)	Notes
MD2Digest	128	
MD4Digest	128	
MD5Digest	128	
RipeMD128Digest	160	basic RipeMD
RipeMD160Digest	160	enhanced version of RipeMD
SHA1Digest	160	
SHA256Digest	256	Draft version from FIPS 180-2
SHA384Digest	384	Draft version from FIPS 180-2
SHA512Digest	512	Draft version from FIPS 180-2
TigerDigest	192	The Tiger Digest.

MAC

Name	Output (in bits)	Notes
CBCBlockCipherMac	blocksize/2 unless specified	
CFBBlockCipherMac	blocksize/2, in CFB 8 mode, unless specified	
HMac	digest length	

Appendix B: CLDC devices

Java-enabled Wireless Devices:

Wireless Technology	Frequency (MHz)	Manufacturer	Model	Java Software	Screen	Available
CDMA	1900	LG Electronics	CX-300L	CLDC 1.0	120x160/8 bits	Yes
CDMA	1900	LG Electronics	Cyber-ez-X1	CLDC 1.0	128x128/2 bits	Yes
CDMA	1900	LG Electronics	I-Book	CLDC 1.0	128x128/2 bits	Yes
CDMA2000 1X	800	Casio	A3012CA	CLDC 1.0, MIDP 1.0	132x176/14 bits	Yes
CDMA2000 1X	800	LG Electronics	C-nain 2000	CLDC 1.0, MIDP 1.0	120x133/8 bits	Yes
CDMA2000 1X	800	LG Electronics	C-nain 2100	CLDC 1.0, MIDP 1.0	8 bits	Yes
CDMA2000 1X	800	Samsung	SCH-X130	CLDC 1.0, MIDP 1.0	128x128/2 bits	Yes
CDMA2000 1X	800	Samsung	SCH-X230	CLDC 1.0, MIDP 1.0	120x160/8 bits	Yes
CDMA2000 1X	800	Samsung	SCH-X250	CLDC 1.0, MIDP 1.0	120x160/8 bits	Yes
CDMA2000 1X	800	Samsung	SCH-X350	CLDC 1.0, MIDP 1.0	128x128/2 bits	Yes
CDMA2000 1X	800	Samsung	SPH-X4209	CLDC 1.0, MIDP 1.0	128x160	Yes

CDMA2000 1X	800	Sanyo	A3011SA	CLDC 1.0, MIDP 1.0	132x176/16 bits	Not yet
CDMA2000 1X	800	Sony Ericsson	A3014S	CLDC 1.0, MIDP 1.0	120x120/16 bits	Yes
CDMA2000 1X	800	Toshiba	A3013T	CLDC 1.0, MIDP 1.0	144x176/16 bits	Not yet
AMPS, CDMA2000 1X	800, 1900	Nokia	3585	CLDC 1.0, MIDP 1.0	95x65/1 bit	Not yet
AMPS, CDMA2000 1X	800, 1900	LG InfoComm	VX1	CLDC 1.0, MIDP 1.0	128x104	Yes
AMPS, CDMA2000 1X	800, 1900	Motorola	T720	CLDC 1.0, MIDP 1.0	120x160/12 bits	Not yet
CDMA	800	Casio	C452CA	CLDC 1.0, MIDP 1.0	120x133/8 bits	Yes
CDMA	800	Hitachi	C3001H	CLDC 1.0, MIDP 1.0	120x162/12 bits	Yes
CDMA	800	Hitachi	C451H	CLDC 1.0, MIDP 1.0	120x143/8 bits	Yes
CDMA	800	Kyocera	C3002K	CLDC 1.0, MIDP 1.0	120x160/16 bits	Yes
CDMA	800	Panasonic	C3003P	CLDC 1.0, MIDP 1.0	132x176/16 bits	Yes
CDMA	800	Toshiba	C5001T	CLDC 1.0, MIDP 1.0	144x176/12 bits	Yes
AMPS, CDMA	800, 1900	Motorola	V60i	CLDC 1.0, MIDP 1.0	96x64	Not yet

GSM/GPRS		Siemens	C55			Not yet
GSM/GPRS	1900	Research In Motion	Blackberry 5810	CLDC 1.0, MIDP 1.0	160x160/1 bit	Yes
GSM/GPRS	850, 1900	Nokia	3590	CLDC 1.0, MIDP 1.0	95x65/1 bit	Not yet
AMPS, GSM/GPRS, TDMA	800, 850, 1900	Sony Ericsson	T62u	CLDC 1.0, MIDP 1.0	101x80/2 bits	Not yet
GSM/GPRS	900, 1800	Motorola	Accompli 008/6288	CLDC 1.0, MIDP 1.0	320x240/2 bits	Yes
GSM/GPRS	900, 1800	Nokia	7650	CLDC 1.0, MIDP 1.0	176x208/12 bits	Not yet
GSM/GPRS	900, 1800	Research In Motion	Blackberry 5820	CLDC 1.0, MIDP 1.0	160x160/1 bit	Yes
GSM/GPRS	900, 1800	Siemens	M50	CLDC 1.0, MIDP 1.0	101x64/1 bit	Not yet
GSM/GPRS	900, 1800	Siemens	SL42			
GSM/GPRS	900, 1800	Sony Ericsson	Z700	CLDC 1.0, MIDP 1.0	96x92/8 bits	Not yet
GSM/GPRS	900, 1800, 1900	Motorola	A388	CLDC 1.0, MIDP 1.0	2 bits	Yes
GSM/GPRS	900, 1800, 1900	Motorola	Accompli 009	CLDC 1.0, MIDP 1.0	240x160/8 bits	Yes
GSM/GPRS	900, 1800, 1900	Motorola	T280i	CLDC 1.0, MIDP 1.0		Not yet
GSM/GPRS	900, 1800, 1900	Motorola	V60i	CLDC 1.0, MIDP 1.0	96x64	Not yet

GSM/GPRS	900, 1800, 1900	Motorola	V66i	CDLC 1.0, MIDP 1.0	96x64	Not yet
GSM/GPRS	900, 1800, 1900	Nokia	6310i	CLDC 1.0, MIDP 1.0	95x65/1 bit	Yes
GSM/GPRS	900, 1800, 1900	Nokia	6610	CLDC 1.0, MIDP 1.0	128x128/12 bits	Not yet
GSM/GPRS	900, 1800, 1900	Nokia	7210	CLDC 1.0, MIDP 1.0	128x128/12 bits	Not yet
GSM/GPRS	900, 1800, 1900	Samsung	SGH-S100		128x160/16 bits	
GSM/GPRS	900, 1800, 1900	Sendo	Z100	CLDC 1.0, MIDP 1.0	176x220/16 bits	
GSM/GPRS	900, 1800, 1900	Sony Ericsson	P800	CLDC 1.0, MIDP 1.0, PersonalJava 1.1.1	320x208/12 bits	Not yet
GSM/GPRS, W-CDMA	900, 1800, 1900	Motorola	A820		176x220/12 bits	
GSM/GPRS, TDMA	800, 900, 1900	Siemens	M46			
GSM/GPRS	800, 900, 1900	Motorola	T720	CLDC 1.0, MIDP 1.0	120x160/12 bits	Not yet
GSM	1900	Nokia	9290 Communicator	CLDC 1.0, MIDP 1.0, PersonalJava 1.1.1, JavaPhone 1.0	640x200/12 bits	Yes
GSM	900, 1800	Nokia	3410	CLDC 1.0, MIDP 1.0	95x65/1 bit	Yes

GSM	900, 1800	Nokia	9210 Communicator	CLDC 1.0, MIDP 1.0, PersonalJava 1.1.1, JavaPhone 1.0	640x200 /12 bits	Yes
GSM	900, 1800	Nokia	9210i Communicator	CLDC 1.0, MIDP 1.0, PersonalJava 1.1.1, JavaPhone 1.0	640x200 /12 bits	Not yet
GSM	900, 1800	Siemens	SL45i/6688i	CLDC 1.0, MIDP 1.0	101x80/ 1 bit	Yes
iDEN	800	Motorola	i50sx	CLDC 1.0, MIDP 1.0	111x100 /2 bits	Yes
iDEN	800	Motorola	i55sr	CLDC 1.0, MIDP 1.0	111x100 /2 bits	Yes
iDEN	800	Motorola	i80s	CLDC 1.0, MIDP 1.0	119x64/ 1 bit	Yes
iDEN	800	Motorola	i85s	CLDC 1.0, MIDP 1.0	111x100 /2 bits	Yes
iDEN	800	Motorola	i90c	CLDC 1.0, MIDP 1.0	111x110 /2 bits	Yes
iDEN	800	Motorola	i95cl	CLDC 1.0, MIDP 1.0	120x160 /8 bits	Not yet
PDC	1500	Mitsubishi	J-D05	CLDC 1.0, MIDP 1.0	12 bits	Yes
PDC	1500	Sharp	J-SH07	CLDC 1.0, MIDP 1.0	120x160 /16 bits	Yes
PDC	1500	Sharp	J-SH08	CLDC 1.0, MIDP 1.0	122x162	Yes
PDC	1500	Sharp	J-SH51	CLDC 1.0, MIDP 1.0	122x162	Yes
PDC	1500	Toshiba	J-T06	CLDC 1.0, MIDP 1.0	16 bits	Yes
PDC	800	Fujitsu	F503i	CLDC 1.0	120x130 /8 bits	Yes
PDC	800	Fujitsu	F503iS	CLDC 1.0	120x130 /10 bits	Yes
PDC	800	Mitsubishi	D503i	CLDC 1.0	132x142 /10 bits	Yes
PDC	800	Mitsubishi	D503iS	CLDC 1.0	132x142 /10 bits	Yes
PDC	800	NEC	N503i	CLDC 1.0	120x130 /10 bits	Yes
PDC	800	NEC	N503iS	CLDC 1.0	120x130 /10 bits	Yes
PDC	800	Panasonic	P503i	CLDC 1.0	120x130 /8 bits	Yes
PDC	800	Panasonic	P503iS	CLDC 1.0	120x130 /8 bits	Yes

PDC	800	Sony Ericsson	SO503i	CLDC 1.0	128x128 /16 bits	Yes
PDC	800	Sony Ericsson	SO503i S	CLDC 1.0	128x128 /16 bits	Yes
PDC	800, 1500	Sony Ericsson	SO504i	CLDC 1.0	128x128 /16 bits	Yes
AMPS, TDMA	800, 1900	Motorola	V60i	CLDC 1.0, MIDP 1.0	96x64	Not yet
W-CDMA		Mitsubishi	D2101V	CLDC 1.0	132x162 /18 bits	Yes
W-CDMA		NEC	N2002	CLDC 1.0	16 bits	Yes
W-CDMA		Panasonic	P2101 V	CLDC 1.0	176x220 /18 bits	Yes

The columns in the above table are as follows:

Wireless Technology and **Frequency** refer to the wireless networks with which the device can communicate. See Table 2 for additional details on these networks, including a sampling of carrier and brand names.

Manufacturer and **Model** are self-explanatory.

Java[tm] **Software** lists the standard software specifications and packages that the device supports. Devices may support additional, nonstandard APIs, but these are not listed here.

Screen describes the device's screen, both its resolution in pixels and its color depth. Color depth refers to the number of bits per pixel that are used for color information. One bit implies straight black and white, while two bits or four bits usually refers to four or sixteen levels of gray, respectively.

Wireless Technologies:

Wireless Technology	Frequency (MHz)	Brand Names	Carriers	Locations
iDEN	800		Nextel, Clearnet, etc.	USA, Canada, Brasil, Israel and Middle East
GSM	900, 1800			Europe, Asia
GSM	1900			North America
GPRS	900, 1800			Europe, Asia
GPRS	1900			North America
CDMA	1900	ez-i	LG Telecom	South Korea

CDMA	800		Shinsegi Telecom, SK Telecom	South Korea		
CDMA2000 1X	800		Shinsegi Telecom, SK Telecom	South Korea		
CDMA	800	ezPlus	KDDI	Japan		
CDMA2000 1X	800	ezPlus	KDDI	Japan		
PDC	800	i-mode	NTT DoCoMo	Japan		
PDC	1500	J-Sky	J-Phone	Japan		
W-CDMA		FOMA	NTT DoCoMo	Tokyo, Japan		

Source: HTTP://wireless.java.sun.com/device/

Appendix C: MIDP1.0 XML-parsers

Name	License	Size	Type
ASXMLP 020308	Modified BSD	6 kB	Push, model
kXML 2.0 alpha	EPL	9 kB	Pull
kXML 1.2	EPL	16 kB	Pull
MinML 1.7	BSD	14 kB	Push
NanoXML 1.6.4	zlib/libpng	10 kB	model
TinyXML 0.7	GPL	12 kB	model
Xparse-J 1.1	GPL	6 kB	model

Source: [Knud02]

Appendix D: MIDlet performance test

All tests are carried out in a Nokia 6310i with a full charged battery. Each MIDlet download were succeeded in less that one minute via GSM.

The same text is used for each encryption: "12345678901234567890123"

The text is a 23-digit number to simulate a standard credit card: 16 digits for card-number, 4 digits for expiring date and 3 digits for control-number.

Several tests are carried out for each cipher. Each test was repeated several times after complete shutdown of the program, but no noteworthy variance was noticed.

The code for Rc6MIDlet:

```
/*
 * RctMIDlet - by Anders Cervera, 11 juli 2002
 * This MIDlet is for testing the speed of a
 * Rc6 cipher in Nokia 6310i
 */

import javax.microedition.io.*;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class Rc6MIDlet extends MIDlet implements CommandListener {

	private Timer timer;
	private long time;
	private Display display;
	private Form inForm;
	private TextField inTextField;
	private Command enterCommand;
	private Command exitCommand;

	private String key;
	private byte[] textToCipher;
	private String finTextCryp;
	private String finTextDecryp;
	private char[] resText;

	public Rc6MIDlet() {
		inForm = new Form("RC6 form");
		inTextField = new TextField("Enter text: ", null, 23, TextField.NUMERIC);
		enterCommand = new Command("Encrypt", Command.OK, 1);
		exitCommand = new Command("exit", Command.EXIT, 1);

		key = "rokijuhyuuhygtrf"; //16byte - 128 bit
		byte[] temp = key.getBytes();
		System.out.println("key length: "+temp.length);
	}
```

```
	public void startApp() {

		inForm.append(inTextField);
		inForm.addCommand(enterCommand);
		inForm.addCommand(exitCommand);
		inForm.setCommandListener(this);

		display = Display.getDisplay(this);
		display.setCurrent(inForm);
	}

	public void commandAction(Command c, Displayable d) {
		if (c == enterCommand) {

			timer = new Timer(); //initiate the timer before encryption

			textToCipher = inTextField.getString().getBytes();
			Rc6 cipher = new Rc6(key, true);

			finTextCryp = cipher.encrypt(textToCipher);

			time = timer.getElapsed(); //get time elapsed after encryption

			Alert a = new Alert("text", finTextCryp + "\n" + "Time elapsed: " + time,
null, null);
			a.setTimeout(Alert.FOREVER);
			display.setCurrent(a, inForm);

			//decrypt again
			Rc6 cipherBack = new Rc6(key, false);
			finTextDecryp = cipherBack.deCrypt(finTextCryp);
			System.out.println("Decryptet text: " + "\n" + finTextDecryp);
		}

		else if (c == exitCommand)
			notifyDestroyed();
	}

	public void pauseApp(){}
	public void destroyApp(boolean u){}

}
-----------------------

/*
 * Rc6 - by Anders Cervera, 11 July 2002
 * This class does the encryption
 *
 */
import java.io.*;
import org.bouncycastle.crypto.BufferedBlockCipher;
import org.bouncycastle.crypto.CryptoException;
import org.bouncycastle.crypto.engines.RC6Engine;
import org.bouncycastle.crypto.modes.PaddedBlockCipher;
import org.bouncycastle.crypto.modes.CBCBlockCipher;
import org.bouncycastle.crypto.params.KeyParameter;
```

```
public class Rc6 {

	private BufferedBlockCipher cipher;
	private KeyParameter symKey;
	private byte[] text;
	private byte[] out;
	private int outSize;
	private char[] charOut;

	public Rc6 (String key, boolean type) {

		//creates the key
		this.symKey = new KeyParameter(key.getBytes());

		cipher = new PaddedBlockCipher(new CBCBlockCipher(new RC6Engine()));

		//initialize the chiper
		cipher.init(type, symKey);

	}

	public String encrypt(byte[] text) {

			this.text = text;

			outSize = cipher.getOutputSize(text.length);
			out = new byte[outSize];

			int oLen = cipher.processBytes(text, 0, text.length, out, 0);
			try	{
				cipher.doFinal(out, oLen);
			}
			catch (CryptoException ce) {
				System.out.println("Encrypt exception: "+ce.toString());
			}

			charOut = HexCodec.bytesToHex(out);
			return new String(charOut);

	}
}
```

```
/*
 * Timer - by Anders Cervera, 11 July 2002
 * This class calculates the time
 *
 */

public class Timer {

	private final long time;

	public Timer () {
		this.time = System.currentTimeMillis();
```

```
	}

	public long getElapsed() {
		return	System.currentTimeMillis() - time;
	}
}


---------------------

/*
 * HexCodec - 11 July 2002
 * This class converts from binary to hex and
 * back
 */

public class HexCodec {

	private static final char[] kDigits = {
		'0', '1', '2', '3', '4', '5', '6', '7', '8', '9',
		'a', 'b', 'c', 'd', 'e', 'f'
		};

	public static char[] bytesToHex(byte[] raw) {
		int length = raw.length;
		char[] hex = new char[length * 2];
		for (int i = 0; i < length; i++) {
			int value = (raw[i] + 256) % 256;
			int highIndex = value >> 4;
			int lowIndex = value & 0x0f;
			hex[i * 2 + 0] = kDigits[highIndex];
			hex[i * 2 + 1] = kDigits[lowIndex];
		}
		return hex;
	}

	public static byte[] hexToBytes(char[] hex) {
		int length = hex.length / 2;
		byte[] raw = new byte[length];
		for (int i = 0; i < length; i++) {
			int high = Character.digit(hex[i * 2], 16);
			int low = Character.digit(hex[i * 2 + 1], 16);
			int value = (high << 4) | low;
			if (value > 127) value -= 256;
			raw[i] = (byte)value;
		}
	return raw;
	}

	public static byte[] hexToBytes(String hex) {
		return hexToBytes(hex.toCharArray());
	}
}
----------------------

/*
```

```
 * DesCryp - by Anders Cervera, 11 July 2002
 * This class does the encryption
 *
 */
import org.bouncycastle.crypto.*;
import org.bouncycastle.crypto.engines.*;
import org.bouncycastle.crypto.modes.*;
import org.bouncycastle.crypto.params.*;

public class DesCryp {

		private BufferedBlockCipher cipher;
		private KeyParameter key;
		private byte[] text;
		private byte[] out;
		private int outSize;
		private char[] charOut;

		//constuctor, initiates the key and ciper
		public DesCryp (String inkey, boolean type) {
			this.key = new KeyParameter(inkey.getBytes());
			cipher = new PaddedBlockCipher(new CBCBlockCipher(new
DESEngine()));
			cipher.init(type, key);
		}

		//method to encrypt the text
		public String enCrypt(byte[] text) {
			this.text = text;

			outSize = cipher.getOutputSize(text.length);
			out = new byte[outSize];

			int oLen = cipher.processBytes(text, 0, text.length, out, 0);
			try {
				cipher.doFinal(out, oLen);
			}
			catch (CryptoException ce) {
				System.out.println("Encrypt exception: "+ce.toString());
			}

			//converts bytes to hex
			charOut = HexCodec.bytesToHex(out);
			//returns the encrypted text to midlet
			return new String(charOut);
		}

}
-------------------------

/*
 * Rc4 - by Anders Cervera, 11 July 2002
 * This class does the encryption
 *
 */
import java.io.*;
```

```
import org.bouncycastle.crypto.StreamCipher;
import org.bouncycastle.crypto.engines.RC4Engine;
import org.bouncycastle.crypto.params.KeyParameter;

public class Rc4 {

	private StreamCipher cipher;
	private byte[] plainText;
	private byte[] cipherText;
	private char[] hexCipherText;
	private byte[] symKey;
	private String deCryp;

	public Rc4 (String key, boolean type) {

		//creates the key
		this.symKey = key.getBytes();
		cipher = new RC4Engine();
		//initialize the chiper
		cipher.init(type, new KeyParameter(symKey));

	}
	//method to encrypt the text
	public char[] encrypt(byte[] text) {

		plainText = text;
		cipherText = new byte[plainText.length];
		cipher.processBytes(plainText, 0, plainText.length, cipherText, 0);
		//converts bytes to hex
		hexCipherText = HexCodec.bytesToHex(cipherText);

		//returns the encrypted text to the MIDlet
		return hexCipherText;
	}
}
-------------------

/*
 * DesedeCryp - by Anders Cervera, 11 July 2002
 * This class does the encryption
 *
 */
import org.bouncycastle.crypto.*;
import org.bouncycastle.crypto.engines.*;
import org.bouncycastle.crypto.modes.*;
import org.bouncycastle.crypto.params.*;

public class DesedeCryp {

		private BufferedBlockCipher cipher;
		private KeyParameter key;

		private byte[] text;
		private byte[] out;
		private int outSize;
		private char[] charOut;
```

```
		public DesedeCryp (String inkey, boolean type) {
			this.key = new KeyParameter(inkey.getBytes());

			cipher = new PaddedBlockCipher(new CBCBlockCipher(new
DESedeEngine()));
			cipher.init(type, key);
		}

		public String enCrypt(byte[] text) {
			this.text = text;

			outSize = cipher.getOutputSize(text.length);
			out = new byte[outSize];

			int oLen = cipher.processBytes(text, 0, text.length, out, 0);
			try	{
				cipher.doFinal(out, oLen);
			}
			catch (CryptoException ce) {
				System.out.println("Encrypt exception: "+ce.toString());
			}

			charOut = HexCodec.bytesToHex(out);
			return new String(charOut);
		}
}
```

Appendix E: Questions for interviews

All questions where tailored each company. Below is a summary of all questions:

Basic questions for all

For introduction:

-What is your company's strategy in relation to m-payments (or internet-payments) and which concrete systems have you been involved in?

-In what directions do you think the market for m-payments will turn - and why?

- Which kind of payment service providers?
- Which front-end technologies?

-Going through the different technologies for connectivity:

- SIM Application Toolkit (SAT), WAP-browser, Java application, SMS/MMS, VOICE, (Others?)
- Discussion of the different strengths/weaknesses
- Discussion of GUI, usability, security etc.

-Do you believe in WAP, SAT, J2ME™ for m-payments in the future?

- Why / why not?

-What is the strength/weaknesses of: mPay, Premium Rate SMS, Wallet, J2ME™?

- Why and what can be better?

-How would you design the optimal m-payment system and why?

-What kind of bearers for m-payment do you believe on - and why?

- Preliminary discussion of CSFs. Presentation of the initial list of CSFs. Do you agree? Why/why not? Which are missing?

For the manufacturers

-What other possibilities are present for storing data from a MIDlet (besides the record store)? (SIM, flash memory?)

-Do you expect to use/implement other kind of persistent storage than the MIDlet record store?

-What efforts does it take to store data in e.g. SIM or other storage facilities from a MIDlet?

-What possibilities are present for enabling secure connections with a MIDlet?

-Which additional API's and features does your phones support- and how?

-What are the prospects of pushing a MIDlet to the phone?

-What are the prospects of pushing a URL to the phone?

-Do you plan to implement the JSR120 (wireless API, SMS etc.)?

-Do you plan to (or have done) implement other "home-made" J2ME™ API's just for your phones?

Questions for consultancy/developers

-Is it possible to crack a MIDlet and the belonging record store to read the content?

-How would you describe the possibilities/suitability of J2ME™ in relation to building an m-payment application?

2.round of interview - for all

-Presentation of the final CSFs and the corresponding requirements.

- Do you agree? Why/why not? Which are missing?

Appendix F: Prototype of Payment MIDlet

Note! This MIDlet is only a prototype and will not work either in a mobile device or in a J2ME™ toolkit simulator. The MIDlet follows the MIPD2.0 API Draft7 for HTTPS connections and would in theory work in mobile phone with the corresponding implementation of the HTTPS. The MIDlet only contains the most simple and basic functionalities to act as a payment terminal. Merchant and order data is hard-coded why the MIDlet does not consider the initial process of payment initiation - The "real" corresponding payment MIDlet would therefore likely contain additional code. These data may e.g. be pushed into the MIDlet via e.g. SMS or bluetooth.

The source code:

```
/*
 * SslTest - by Anders Cervera, 14 juli 2002
 * This MIDlet is a prototype for HTTPS connections to Architrade payment systems
 *
 */
import java.io.*;
import javax.microedition.midlet.*;
import javax.microedition.io.*;
import javax.microedition.lcdui.*;


public class SslTest extends MIDlet implements CommandListener {

        private final String merchant="4181607";
        private final String amount="19095";
        private final String currency="208";
        private final String orderid ="991002b";
        private final String accepturl ="https://payment.architrade.com/cgi-
ssl/relay.cgi/http://www.java4mobile/dibs/godkendt.jsp";
        private final String declineurl ="https://payment.architrade.com/cgi-
ssl/relay.cgi/http://www.java4mobile/dibs/afvist.jsp";
        private final String test="foo";

        private String cardno;
        private String expmon;
        private String expyear;
        private String cvc;
        private String url = "https://payment.architrade.com/cgi-ssl/auth.cgi";
        private String post;
        private String urlTotal;

   private Command exitCommand  = new Command("Exit", Command.EXIT, 2);
   private Command getCommand = new Command("Pay", Command.SCREEN, 1);
   private Form form;
```

```
        private TextField txtCardno = new TextField("Card no:", null        , 16,
TextField.NUMERIC);
        private TextField txtExpmon = new TextField("Expmon", null, 2, TextField.NUMERIC);
        private TextField txtExpyear = new TextField("Expyear", null , 2, TextField.NUMERIC);
        private TextField txtCvc = new TextField("Cvc", null, 3, TextField.NUMERIC);

    private Display display;

    public SslTest() {
    }

    public void startApp() {
                if (display == null)
                        display = Display.getDisplay(this);

                form = new Form("Payment");
                form.append(txtCardno);
                form.append(txtExpmon);
                form.append(txtExpyear);
                form.append(txtCvc);
                form.addCommand(exitCommand);
                form.addCommand(getCommand);
       form.setCommandListener(this);
                display.setCurrent(form);
    }

    public void commandAction(Command c, Displayable d)  {

                if (c == exitCommand) {
                        notifyDestroyed();
                }
                else if (c == getCommand) {

                        cardno=txtCardno.getString();
                        expmon=txtExpmon.getString();
                        expyear=txtExpyear.getString();
                        cvc=txtCvc.getString();

                        post =
"?merchant="+merchant+"&amount="+amount+"&currency="+currency+"&orderid="+orderid+
"&accepturl="+accepturl+"&declineurl="+declineurl+"&cardno="+cardno+"&expmon="+expmon+
"&expyear="+expyear+"&cvc="+cvc+"&test="+test;

                        StringBuffer b = new StringBuffer();
                        HttpsConnection con = null;
                        InputStream is = null;
                        OutputStream os = null;

                        urlTotal = url+post;

                        try {
                           int len = 0;
                           int ch = 0;
                        con = (HttpsConnection)Connector.open(urlTotal);
                        con.setRequestMethod(HttpsConnection.POST);
```

```
                    /*
                byte[] data = post.getBytes();
                        con.setRequestProperty("Content-Length",
Integer.toString(data.length));
                        os = con.openOutputStream();
                        os.write( data );
                        os.close();
                        */

                        System.out.println(Integer.toString(con.getResponseCode()));

                    is = con.openInputStream();
            len = (int) con.getLength();

            if (len != -1) {
              for(int i=0; i<len; i++) {
                if((ch = is.read()) != -1) {
                  b.append((char) ch);
                }
              }
            }
            else {

              while((ch = is.read()) != -1) {
                len = is.available();
                  b.append((char) ch);
              }
            }

                        System.out.println("Response: " +b.toString());
                        Alert a = new Alert("Trans results:", b.toString(), null, null);
                        a.setTimeout(Alert.FOREVER);
             display.setCurrent(a);


                } catch (Exception e) {
              e.printStackTrace();
              String s = e.toString();
              if(s != null) {
               Alert aa = new Alert("Error in connection:", s, null, null);
                aa.setTimeout(Alert.FOREVER);
                display.setCurrent(aa);
              }

            } finally {
              if (is != null) {
                    try {
                        is.close();
                    } catch (Exception ce) { }
              }
              if (c != null) {
                    try {
                        con.close();
                    } catch (Exception ce) { }
              }
            }
```

```
                    }
    }

    public void pauseApp() {
    }
    public void destroyApp(boolean unconditional) {
    }
}
```