Usability and security in a messaging prototype for mobile phones

Lars Mikkel Aas

Master's Thesis
Master of Science in Information Security
30 ECTS
Department of Computer Science and Media Technology
Gjøvik University College, 2007

Avdeling for
informatikk og medieteknikk
Høgskolen i Gjøvik
Postboks 191
2802 Gjøvik

Department of Computer Science
and Media Technology
Gjøvik University College
Box 191
N-2802 Gjøvik
Norway

Abstract

Messaging through the GSM mobile network is not particular secure. This thesis investigates usability issues related to the look'n'feel and the user interface of a secure SMS service on a range of mobile phones.

We have carried out a field test of a prototype for secure SMS messaging. A total of 21 users participated in the field test, which did last for 32 days. Additionally we conducted a survey among the participants after the field test was completed. Our findings relates to Whitten and Tygar's definition of usable security software used in their article *Why Johnny can't encrypt*.

Our study shows that by making the security as transparent as possible from the users perspective, the users quickly learn how to operate the messaging system efficiently without making many errors.

Sammendrag - Abstract in norwegian

SMS meldinger i dagens GSM nettverk er ikke a betrakte som sikre. Denne masteroppgaven vil undersøke spørsmål knyttet til brukeropplevelser av en prototype for sikre meldinger på mobiltelefoner.

Vi har gjennomført et felt-forsøk av en prototype for sikre SMS meldinger. 21 personer deltok i forsøket, som varte i 32 dager. I tillegg gjennmførte vi en spørreundersøkelse blant deltagerene i etterkant av felt-forsøket. Vi vil knytte våre funn til Whitten og Tygar's definisjoner brukt i deres artikkel *Why Johnny can't encrypt*.

Vårt arbeid viser at ved å gjøre sikkerhetsmekanismene så usynlige for brukeren som mulig, så vil brukerne fort lære seg å håndtere applikasjonen for sikre meldinger på en effektiv måte og uten å gjøre for mange feil.

Contents

List of Figures

List of Tables

Acknowledgements

I would not have been able to do this thesis without help. First of all I would like to thank supervisor Einar Snekkenes, professor at Norwegian Information Security Laboratory (NISLAB) at Gjøvik University College.

I must also give thanks to all my colleagues — no one mentioned, no one forgotten. I would also thank the staff at Gjøvik University College library for great service.

Finally; a great thank you to all my fellow students, for a fun and educational time together.

—Lars Mikkel Aas

1 Introduction

1.1 Topic covered by this thesis

This thesis investigates the extra effort needed to be able to send and receive secure messages on a java enabled mobile phone, and how users learn to use it. The thesis will look at two conflicting dimensions; usability and security in the mobile domain. This is done by doing a field test of a prototype for encrypted SMS messages.

Keywords: Usability vs Security, J2ME, usability engineering

1.2 Problem description

As is, SMS using a standard mobile phone is not particular secure [1]. In an electronic mail setting there are many different systems for secure communications. But often an increased level of security results in a bad affect on the user experience, in terms of usability and efficiency. Whitten and Tygar [2] has shown that in a stationary setting the user found it difficult to operate the messaging software in a secure way.

1.3 Justification and motivation

In Norway there are more mobile phones than inhabitants (104% coverage) and over 5 billion SMS's was sent during 2006 [3]. Until recently SMS messaging have been rather innocent and trivial in it's use, but more and more of our business communications are based on SMS messages. Even the security systems at banks [4] and intranets utilize SMS messages as part of their authentication process. The amounts of SMS messages sent increases, and with it, the range of use expands, but services to secure them are almost non-existing.

The Norwegian Data Inspectorate [5] have also given some guidelines on cryptographic strength when transmitting personal data in open data- and communacation network, like the GSM networks. In 2001 they recommended a cryptographic strength equal to DES128 (112 bits effective) or stronger, and emphasized that one should increase this level from time to time, due to the increased level of computing powers available for attacks.

It is not likely that ordinary mobile phones will be used by government or military as high-grade crypto devices. Those organizations have the resources to manage this on their own. Such devices are expensive, often have a big and clumsy form factor, intended for one purpose only, and finally there are usually strict policies regulating the use of them.

On one side we have the expensive high-grade crypto devices, and on the other we have the ordinary mobile phone. With this in mind, small, affordable common devices, such as mobile phones, can fill the gap between the high grade crypto systems described above and the insecure services of the GSM network. This way one can achieve affordable, medium security with good usability.

1.4 Who will benefit?

A thorough and documented research in this field, together with a functional prototype, will provide useful knowledge on how to specify, design and implement a secure messaging solution with mobile phones. When designing and planning for high grade secure messaging services, the lessons learned from our prototype can contribute to create specifications and deliver valuable experience in a quick, easy and affordable way.

The stakeholders are people and organizations who need a simple, affordable and user-friendly secure messaging system. In this thesis' case it is specific Police departments in Oslo, Norway who will benefit the most.

1.5 Research questions

This thesis is inspired by the article *Why Johnny can't encrypt* which was published in 1999 [2]. In this article the users, a.k.a. "Johnny", tries to communicate securely in a stationary environment, using electronic mail and PGP 5.0 on a personal computer. Today, in 2007, Johnny communicates with SMS messages on his mobile phone. Are Whitten and Tygar's findings relevant in a mobile setting too?

- Which solutions available today offer secure messaging on mobile phones, and what are their costs, pros and cons?
- Even with the goal of making security transparent in the software, the user will have to perform a minimum of security related tasks. How well will Johnny perform those tasks?
    - Will he get comfortable with the software?
    - How fast will he learn the software?
    - Will he make errors?

1.6 Scope

Due to limited time and resources this thesis has not tried to invent and deploy a perfect secure messaging system. Nor have we focused highly on security engineering of the prototype. This thesis has focused on the usability aspects of secure messaging, the trade-offs the users will have to make and the impact of them. The prototype offered security look'n'feel rather than proper built in security.

Regarding the choice of equipment and technology it became evident early on that ordinary java-enabled mobile phones was our target platform; Java 2 Micro Edition (J2ME) was an obvious and suitable choice for this thesis.

1.7 Summary of claimed contributions

Whitten and Tygar's article *Why Johnny can't encrypt* has shown that in a stationary setting the users found it difficult to operate the messaging software in a secure way. This thesis shows that, in a mobile setting, it is possible to hide much of the security technology in such a way that the user interface presented to the user is acceptable, thus creating an efficient user experience with a satisfying level of security.

2 Related work

In this chapter we will provide a overview of existing secure messaging solutions available today. Then we will take a look at the state of the art regarding usability insepction methods. The *Handbook of usability testing* by Jeffrey Rubin [6] have been our valuble companion throughout the entire master's thesis.

2.1 Existing solutions

There are many manufacturers that offers secure mobile phones. The Rohde & Schwarz [7] modified Siemens S35 (TopSec GSM), NSK 200 [8] and Global-Teck [9] are good examples of existing solutions. In figure 1, 2 and 3 one can see pictures of them.

That was just a small sample, and numerous more systems do exist, but they all share a couple of common properties:

Expensive Most solutions, and especially those targeting government and military custumors don't mention anything about purchase prices or operational costs. Given the authors experience with similar systems, and the fact that no one claim to be reasonably priced, one can assume that they are expensive.

Speech-oriented Until recently, secure mobile solutions have mostly been speech-oriented, but some offers secured GPRS connections as well. Some of the newer models support SMS.

Old technology The secured phone models are rather old. Siemens S35 for instance, was introduced on CeBIT back in 2001. Using one of these models in public places could attract unwanted attention.

High grade design A large number of solutions aim toward government-, defense- or law enforcement-customers, and are therefore not suitable for small organisations or ordinary citizens.

Figure 1: TopSec GSM

Figure 2: Global-Teck

Figure 3: NSK 200

Proprietary Few vendors tell how their solutions work or how their cryptography are implemented. This is not a good thing.

Management Some solutions tend to be rather cumbersome and labor-intensive to manage.

One purpose From a users perspective, one phone is all he or she needs. To introduce another device just for secure calls and messages will be met with considerable scepticism. Users like to have the newest and hottest phone available, and the old, but secure, phones will loose.

And, since so few of the vendors talk about their implementations of crypto algorithms, key-lengths, key management, and so on, how can one trust them? Schneier [10] says that when it comes to the choice of cryptography, one should use an algorithm that is publicly available, unless you have the resources, to make them your self.

There are of course ways to make sure the system is trustworthy. One can, for instance, get it tested and certified according to Common Criteria [11]. NIST [12] or the respectively national security authority, like the Norwegian NSM [13], can also evaluate products and applications. This is a costly and time-consuming task, and there are always assumptions made in these test, that don't always reflect reality. Microsoft Windows 2000 and it's Common Criteria EAL 4+ certification is a good example.

On Windows Mobile, with PocketOutlook, Microsoft supports integration with Exchange-server[1], and with this comes the ability to use certificates for encryption and signing of electronic mail [14]. On top of this comes Direct Push, which adds push-functionality. This solution might be good for businesses and organizations that already have Exchange Server 2003 and Certificate Server within their organization. Otherwise, this solution is rather expensive, and works only on Windows Mobile devices.

At the time this thesis was written The Directorate for Emergency Communication in Norway was building a new digital radio communications system for the different

[1] this requires Messaging and Security Feature Pack and Exchange Server 2003 Service PAck 2

public safety authorities(police, fire brigade and ambulance service) [15]. It's based on TETRA [16] and is mostly a speach oriented system, but it also has data- and messaging-capabilities. TETRA, however, will not be able to solve the problems this thesis addresses. It is a very expensive, high-grade, government funded system and it requires additional handsets. In addition, the messaging and data capacities are limited.

2.2 Secure messaging models

> The term secure messaging refers to the ability to provide data confidentiality, data integrity, data origin authentication, and non-repudiation of origin services for email.
>
> —Rolf Oppliger [17]

The key factor for a secure messaging system, is it's ability to withstand attacks. There are several implementations of secure messaging for e-mail, were PGP and S/MIME is the most acknowledged and widespread. PGP has been around for some time, and is viewed upon as well designed and engineered. But, as Whitten and Tygar [2] shows, it has usability problems which allow the users to make dangerous errors.

It may look like these kinds of models/systems tries to solve all problems at once, struggling to be perfect. This may not always be what the users need. In a military setting, this would mean that all info has to be considered as "Top Secret". But, we all know that most data don't require the top-most grade or protection, as this is both costly and labor intensive.

The problem in many secure messaging schemes, including electronic mail, is the dependence on a widespread public key infrastructure (PKI) [18]. This is one of the areas where PGP failed [2], as the administration of keys is completely up to the user; he/she has to publish the right keys, do backups on so forth.

Oppliger [17] talks theoretical about the different types of secure messaging schemes. Basically there are these types of systems:

- Systems that require a Trusted Third Party (TTP)
  - Inline TTP
  - Online TTP
  - Offline TTP
- Systems that don't require a Trusted Third Party (TTP)
  - Systems based on simultaneous secret exchange
  - Systems based on trusted systems

Oppligers conclusion states that an online TTP would be the best approach for a certified e-mail scheme on the Internet, as opposed to an inline TTP which seems to be the already established services today.

Roth [18] is proposing a better system for secure and usable electronic mail systems. Among the most interesting findings showed, is that one should not try to explain the unexplainable, such as advanced cryptography, and that the security mechanisms should operate transparent.

Grinter and Smetters [19] use the phrase implicit security, and it is a part of their 3 challenges for embedding security into applications. Here they talk about how to infer security actions based on the users intent. This is not an easy task, and there is not presented any proposed solutions to the challenges.

2.3 Security and devices

Software for devices doesn't differ much from other software, in terms of how to make the software secure. Viega and McGraw's 10 principles of software security [20] is still highly relevant, even for small applications in the mobile world. Yee [21] also states guidelines for secure interaction design, and together with Viega and McGraw they agree that security or usability isn't something one can bolt on to the application at the end of the process.

Nevertheless, there are issues regarding mobile applications that need to be addressed properly, especially because of smaller screens, less CPU power, connectivity, battery-lifetime and so on. When you design mobile and embedded applications there are some special challenges that must be addressed. Grinter and Smetters [19] speaks of some of these challenges regarding how and when to embed the security into the application, thus hiding it and spare the user from making too many security related decisions. There are also new and different challenges to tackle when designing for mobile devices. Raghunathan et al [22] speaks of some of them, like the fact that devices is lost or stolen, their battery-lifetime, less CPU power and the smaller physical form factors.

2.4 Usability - why is it important?

Why is usability important? First of all, it has been a way of gaining market shares. As technical gadgets get more and more complicated the need of good usability has increased. In the mid- to late 1990's Nokia achieved very good market shares due to design and usability [23]. In a commercial perspective, good usability is crucial. Maguire [24] looks at how usability would effect on the attractiveness of products and systems. Even though usability don't beat *functions* and *style* in terms of why people like things, usability seems to be a key factor to why people doesn't like things. In practice this means that usability is a key factor to why people don't buy a particular brand or make again. Regarding mobile phones this will make a difference, as people buy knew phones relatively often.

The security software market is different, and it has not been too focused on the usability aspect. Here the quantitative measures have been the most important; like strength of the encryption, performance and so on.

Good usability is important in many areas, not only regarding security or gadgets. One proof of how usability can be a major player in most systems is given in the book *The human factor : revolutionizing the way people live with technology* where Vicente [25] talks about how bad usability influenced the outcome of the presidential election in the United States of America in 2000.

Mobile phones and SMS messaging are mostly used by the younger part of the population. Soriano et al [26] have done research on how middle-aged users, 35-60 years of age, get along with SMS messaging in general. Their findings shows that middle-aged user experience usability related problems with SMS messaging, and that they should be considered when designing user interfaces on mobile phones.

2.5 Usability - engineering and inspection

Usability inspection is a generic term for a range of usability engineering methods that have seen explosive growth since the first two, heuristic evaluation and cognitive walkthrough, were formally presented at lectures during the ACM CHI'90 conference...

—Jakob Nielsen [27] (Preface)

In the early 1990s specific methods on how to do usability inspection (aka usability engineering, user interface evaluation etc) was developed and presented [28] [29]. Since then methods have changed, adapting to the development of new technology; some have been discarded and new methods have seen the light of day. In [27] Nielsen summarizes and explains the most used usability inspection methods.

Lim [30] et al look at 3 different types of usability evaluations on mobile phones:

Prototype This is a fully functional application, running on it's intended hardware. The look'n'feel is realistic.

Computer-based This is a computer-based low-fidelity prototype. Often it is an emulator running on a PC. It looks pretty much like the intended piece of hardware, but one can not achieve the "feel".

Paper-based This resembles a puppet-theatre, where the look and behavior of the "software" is done through paper and cardboard.

Lim et al's shows that the fully functional prototype is the best way to find usability related issues.

There are basically two ways of conducting an evaluation with a fully functional prototype; a laboratory test or a field test. The laboratory test takes place in s controlled environment, where the user is observed and don't get disturbed or influenced by other and uncontrolled factors. The field test takes place in a real every-day setting. Duh [31] looks at the differences between these two settings when they evaluate usability issues on a mobile phone. Their findings show that a field test is a better way to find usability problems on mobile phones.

2.6 Usability - safety, security and privacy

Given a choice between dancing pigs and security,
users will pick dancing pigs every time.

—McGraw & Felten [32] (Chapter 1, Part 7)

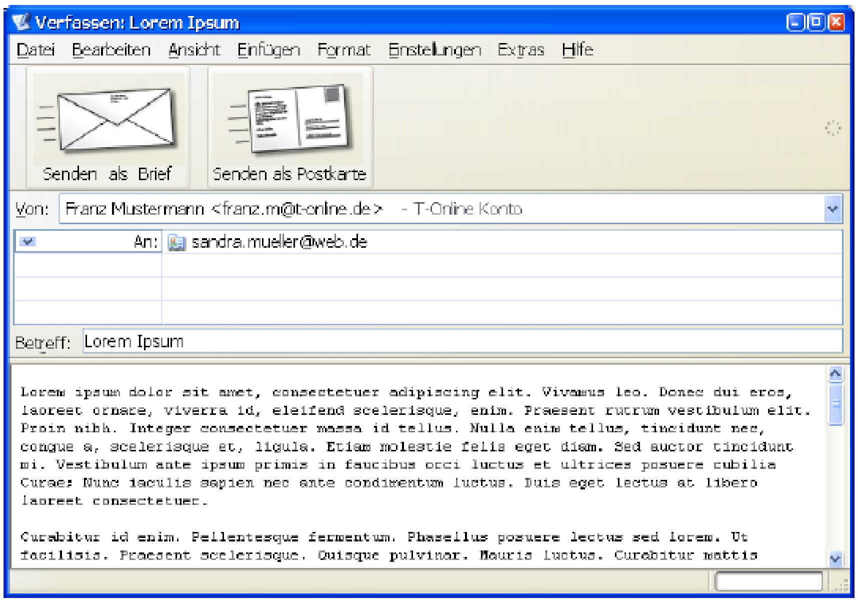
Before computers became common like today, usability was something one was concerned with when planning and designing control centers, e.g. at a power plant, factories or traffic controls. The disaster at Three Mile Island Nuclear Power Plant in 1979 is often mentioned as one of the first good (!) examples of what a bad user interface can lead to [33].

Just as control centers don't provide safety if the users don't know to operate them, security software isn't particular secure if the users makes dangerous errors; a high-grade solution may be impossible to break, but it doesn't really matter if the users don't use it correctly. With this in mind, one has begun to look at usability and security as two related measures. Roth [18] speaks of 80/20 security - based on the assumption that the last 20% of a project require 80% of the efforts. It origins from the fact that many security systems seem over-engineered and designed for an ideal world. Therefore, Roth asks: Can we have 80% security with 20% of the effort? Smetters [34] asks the question differently and wonders *putting usability first, how much security can we achieve?*

In [18] Roth talks mostly about security and usability regarding electronic mail. There he concludes the following:

- The design should be as simple and small as possible
- The mechanisms should have fail-safe defaults
- The mechanisms should be easy to understand and use

These 3 points can hardly be viewed upon as particularly astonishing, but that's not the point either. In their approach the security-part of the E-mail solution should be transparent to the user. In practice, and this is one of their good contributions, they made an e-mail application with two different send-buttons; Send as letter and send as postcard. See figure 4. The security is hidden (transparent) and the user is given two choices he/she can easily relate to.



Von: Franz Mustermann <franz.m@t-online.de> - T-Online Konto

An: sandra.mueller@web.de

Betreff: Lorem Ipsum

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Vivamus leo. Donec dui eros, laoreet ornare, viverra id, eleifend scelerisque, enim. Praesent rutrum vestibulum elit. Proin nibh. Integer consectetuer massa id tellus. Nulla enim tellus, tincidunt nec, congue a, scelerisque et, ligula. Etiam molestie felis eget diam. Sed auctor tincidunt mi. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Nunc iaculis sapien nec ante condimentum luctus. Duis eget lectus at libero laoreet consectetuer.

Curabitur id enim. Pellentesque fermentum. Phasellus posuere lectus sed lorem. Ut facilisis. Praesent scelerisque. Quisque pulvinar. Mauris luctus. Curabitur mattis

Figure 4: Send as letter or Send as postcard

Microsoft TechNet has made 10 immutable laws of security administration [35]. They are practically oriented and technology independent. Law no. 2 reads that *security only works if the secure way also happens to be the easy way* [35].

In the book *Security and Usability: Designing secure systems that people can use* Cranor [36] have a thorough walk through the different aspects of usability and security. Through cases and discussions one get a good overview of authentication methods, phishing, PKI, privacy among other things - all with a usability perspective.

Poor usability is listed as one of *19 deadly sins of software security* [37]. There Howard [37] look at how to spot the sin, and give good advices on how to both avoid and fix them.

They also claim that one of the key principles of building usable, secure software is that *developers are not users*. Their second principle, *security is (almost) never the user's priority*, are more questionable. The word "almost", makes them get away with it. Although the principle is suitable for most applications, it isn't the case in a secure SMS system. Here the security is exactly what the user prioritizes; otherwise he or she would send a regular SMS.

Even though usability many times seems to be the loosing part when it comes to security, other important areas may suffer if one increase usability without concerns about other major areas. Historically this has been the case; increased security gives decreased usability, and vice versa. Lately there have been some concerns about how privacy can suffer due to usability priorities [38].

3 Choice of methods

In this chapter we will discuss which methods we found suitable for our needs; that is, methods useful for finding answers to the research questions. The chapter constist of sections corresponding to the research questions. We have used a mixed approach, utilizing several methods suited to find answers to the different problems at hand.

3.1 Finding existing solutions

In order to find existing systems offering secure messaging on mobile phones, there were two suitable methods. First, a literature study would provide good theoretically background. Databases like ACM Portal[1] and IEEExplore[2] proved to be very useful. Additionally, the Gjøvik University College library, and the staff there, provided a good amount of relevant information.

Secondly, regular Internet search engines, like Google[3] were used, especially to get an overview of available products and solutions, and their specifications. Google Scholar[4] was also used. The authors colleagues at the Police department and other acquaintances also delivered useful information about existing solutions.

[1]http://portal.acm.org/portal.cfm
[2]http://ieeexplore.ieee.org/
[3]http://www.google.no/
[4]http://scholar.google.no/

3.2 Making security transparent in a secure messaging prototype

First we will talk about the process towards finding answers to the research questions. Below are an intro and a brief description of every part of the process. Later, we will discuss key elements more thoroughly.

Literature study Again, a trip to the library was useful. Here one can find relevant books and access different research databases. See previous section for more details.

Field test There are many ways of doing usability inspections, but we chose to do a field test. Literature found during the literature study supports this. It also became evident that we needed a piece of software that actually offered some kind of secured messaging capabilities. We found no one suitable for our needs, and decided to build a prototype.

- **Prototype** The prototype was developed and given a set of features. We did not aim for a perfectly secure and robust system. The goal was to provide security look'n'feel, with as transparent security as possible. See section 4.
- **Usage logging** One of the reasons for making a prototype our self was to be able to log activity - that is, each user's use of the software during the field test.
- **Frustration** A special feature called Frustration was built into the prototype. Here the users could state their level of frustration regarding the prototype its functions and general behaviour.
- **User manual** A small user manual was produced and handed out to the participants of the field test. The purpuse was to give the users a minimum of knowledge on how to use the prototype. See appendix A.

Preliminary analysis Just after the field test was done; all the logs were collected and inserted into a database. Then a preliminary analysis was conducted in order to make good and relevant questions in the following questionnaire.

Questionnaire A usage log analysis can provide great data about usability issues in an application. However, it may also raise new questions or leave other questions unanswered. A questionnaire can provide additionally insight.

Indicators To be able to organize and present the results form the analysis; we needed to define a set of indicators or measures. More about this later, see section 3.2.6.

Analysis Finally, a thorough analysis of both usage logs and questionnaire answers will ultimately provide answers to our research questions.

3.2.1 Field test

Based on the findings of Lim [30] and Duh [31] it was clear that a fully functional prototype field test was the best choice. Nemeth [39] categorizes this as a *validation* usability test. Charlton [40] referres to it as a *experemental testing*, and in table 1 Nemeth's [39] properties for this method is listed. Here one can see why a validation type test is suitable for this thesis.

Description	Users perform a task using a prototype and performance measures are recorded.
Benefits	Because this is often a disassociation between subjective preference and human performance, this type of testing is most accurate.
Disadvantages	High cost, high effort and time-consuming; the focus of the experiment test may be so narrow that it may not be cost effective.
When to use	When given enough resources and when the consequences of human error is high.

Table 1: Nemeth's properties of a validation test

Validation type test (aka experimental test) is obviously a good choice for this thesis' approach. Nielsen [41] (section 7.6) seems to support this choice. But, this method alone may not be enough to address all the problems at hand, so additional method(s) should be used in combination, hence the questionnaire.

3.2.2 Usage logging

When doing usage logging it is crucial that the logging doesn't affect the performance or stability of the system. The logs themselfs should also be unaffected of any failures the prototype may have. In our prototype, the built-in Record Management Store (RMS) [42] record store was used for storing the log data. This record store is reliable, efficient and most important of all, it is persistant. It is record based, and this makes it suitablet for a log-system, storing one log entry in one record in the record store. One thing to look out for, though, is the limited memory available. One should keep the events to log at a minimum. More about this are found in chapter 4.

3.2.3 Frustration

A special feature in the prototype registered the users frustration level at the current time. The users could log their frustation level from 0 through 5, where 5 is the most frustated. A feature like this could both give us useful data for later analysis, but also to ease the users pain, if any. It feels good to blow of some steam from time to time, and this was a key motivation factor for us regarding this feature.

3.2.4 Choice of population

This thesis recruited its population from both policemen and -women, clerks, engineers and other types of personell within a norwegian police organisation — the authors employer. This does not give a perfect normal distribution amongst the participants. We prioritized to get as many participants as possible, rather than having a Gaussian distribution; those who wanted to participate were allowed so, even if we already had equal participants already.

Nemeth [39] stresses that a good knowledge about the participants are crucial. Age, sex and mobile phone experience are examples of data that was collected about the particpants through the questionnaire. See appendix B.

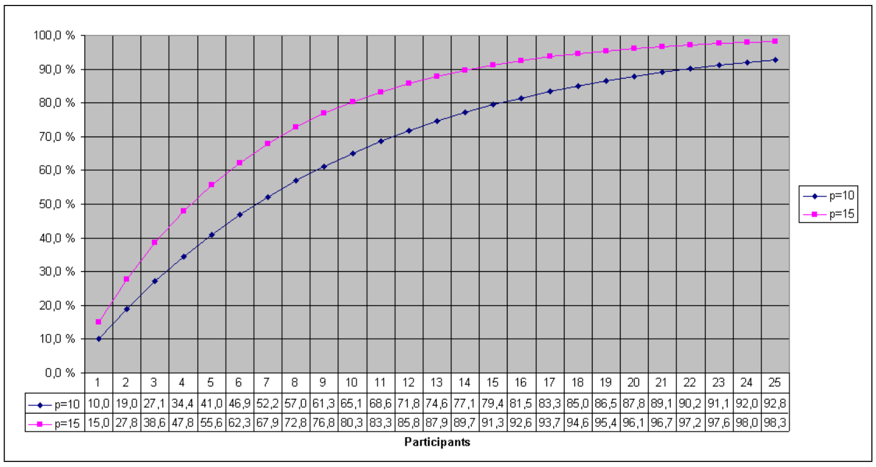
When one looks at other significant usability studies, it strikes one how few participants they use in their tests. Whitten and Tygar [2] had 12 participants and Lim [30] had 15 — 5 for each of their 3 tests. How many participants is needed? Nielsen [41] and Travis [43] debates how to determin the number of participants needed to achieve good results in different usability inspection methods, and Landauer and Nielsen [44] presents an approximation formula to estimate the number of participants needed in usability tests:

$$\text{Usability Problems Found}(i) = N\left(1 - [1 - p]^{i}\right)$$

N = the number of usability problems in the interface,
i = the number of participants and
p = the probability that any single problem will be discovered by any single user

Even with a formula it is difficult to find the exact number of participants needed. The value of *p* is hard to estimate. By setting N = 100 percent and choosing a rather pessimistic value of *p*, *p=10%* - meaning there is a 10% chance that any single usability problem will be discovered by any single participant - we can use the formula to estimate the users needed.

With *p=10%* and 20 users, the formula tells us that 88% of the usability problems will be found. Increasing *p* to 15% gives us 96%. See figure 5 for details.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
p=10	10,0	19,0	27,1	34,4	41,0	46,9	52,2	57,0	61,3	65,1	68,6	71,8	74,6	77,1	79,4	81,5	83,3	85,0	86,5	87,8	89,1	90,2	91,1	92,0	92,8
p=15	15,0	27,8	38,6	47,8	55,6	62,3	67,9	72,8	76,8	80,3	83,3	85,8	87,9	89,7	91,3	92,6	93,7	94,6	95,4	96,1	96,7	97,2	97,6	98,0	98,3

Figure 5: Participants needed in field test

Our approximation doesn't differ much from Nielsen's [41] guidelines. He estimates, when logging actual use in applications, the amount should be at least 20. For a questionnaire one should have at least 30, but this is for a questionnaire-only method.

Based on the approximation formula, the pessimistic p-value and the mentioned guidelines, 20 participants or more will be sufficient.

3.2.5 Built-in obstacles

Our goal was to make the security in our secure messaging prototype as transparent as possible. This is a major challenge, and we implemented only encryption of SMS content, thus making it marginally more secure than regular SMS messages. Nevertheless, there are still some things that can't be totally transparent for the users.

And since we want to see if the users make dangerous errors, we must make it possible for them to do so. The application requires a PIN to launch, and it was programmed to have 9999 as it's default PIN. PIN codes like this, should be changed, and the users also should either be forced or encouraged to do so. We, on the other hand, chose not to do either. This is one of the obstacles we made.

The other obstacle we included into our prototype, was the option to send the SMS message as Unsecure - plain text regular SMS. This is, of course, not something one would not normally do.

3.2.6 Indicators

As mentioned earlier, we will try to match our findings to Whitten and Tygar's [2] article, *Why Johnny can't encrypt*. Since our prototype, on a mobile platform, differs greatly from PGP 5.0 on a personal computer, we will have to create comparable measures. We will have to construct a set of indicators, which we later will analyze against key definitions in Whitten and Tygar's article.

Learning curve

This measure utilizes statistical methods on the recorded time spent on each users messages throughout the experiment, and finds a degree of learning for each user. This indicates if the user has got more effective and familiar with the prototype. The time they spent on typing the actual content of the message is withdrawn. This way short messages and long messages will easily compare.

We look at each users recorded time spent, and calculate a linear trend analysis for each user. It must be emphasized that this linear trend analysis by no means can be used for predictions of any kind; the actual learning curve is not linear, but the users improvement, within the data set from the experiment, can be represented as a linear graph.

Based on the trend line, we can calculate measures for each users improvement.

Awareness

We knew the population consists mostly of policemen and women, who one would think is relatively concerned of security. The security mechanisms built into the prototype were a PIN code and an encryption of the SMS message. The prototype asked for a PIN code in order to launch, and this PIN code could be changed by the user. In the questionnaire we asked some questions regarding the PIN code and if the users could identify the security mechanisms built into the prototype.

Question 6 What kind of security mechanisms were buildt into the application?

Question 7 Did you change the PIN code? If No, state a reason for this.

$$\text{PIN Score} = \frac{n_1}{p}$$

n1 = no. of users who considered PIN a security mechanism
p = no. of participants

$$\text{Encryption Score} = \frac{n_2}{p}$$

n2 = no. of users who considered encryption a security mechanism
p = no. of participants

$$\text{PIN Change Score} = \frac{n_3}{m}$$

n3 = no. of users who changed their PIN
p = no. of participants

Frustation level

This measure will show how the frustration amongst the participants were. When all the registered frustation levels is distributed by date, we can calculate how the frustration levels developed throughout the test period.

Acceptance

This measure tells us how the users used the prototype, considering how many new messages there were composed in relation to the total number of messages sent. If the user composed many new messages compared to replies, this indicates that the prototype was in an active manner. It is important to remember that score values close to 0 or 100 is not desired values. If it is too close to 0 it proves that the users only replied on messages, and had a less active role in the experiment. Values closer to 100 shows that no one replied on messages, which may indicate that the users didn't understand the reply-feature, or that the recipient didn't receive it.

$$\text{New Messages Score} = \frac{s}{t}$$

s = no. of new messages (not replies)
t = no. of total messages, replies included

The next score tells us how active the users were; how many features they used. These values will be calculated per user.

$$\text{Features Score} = \frac{u}{f}$$

u = no. of features used by the users
f = total no. of features

Errors

Here we focus on the actual errors that occures. The prototype will log any errors occuring during run-time. In addition all reports from the users will be manually logged and analyzed. The number of plain-text (regular) messages sent can also contribute to evaluate the level of errors the users made.

$$Plain\ Text\ Messages = \frac{r}{t}$$

r = no. of regular plaintext messages sent
t = no. of total messages, replies included

$$Error\ Score = (c \times 10) + e$$

c = no. of critical errors
e = no. of errors and bugs

4 Prototype

As mentioned in section 1.6, this thesis does not have the luxury of time and resources to implement every aspect of secure messaging. It was necessary to keep the features at a minimum, not only to be able to complete the prototype in time, but also not to over-design the prototype. This way we could make a secure yet usable system.

4.1 Features

The prototype had it's own inbox where the received messages were stored. It was sorted by the received-date of the SMS. Additionally, these features were programmed into the prototype, and available for the test-users:

PIN The user must enter a PIN to launch the application. The user has 3 attempts to, get this right, or else the application exits and the user must start all over again.

Send SMS The main purpose of the prototype is the ability to send messages, both regular SMS and encrypted SMS.

Receive encrypted SMS The prototype can only receive encrypted messages. Once received, they are decrypted and stored in the inbox of the prototype.

PIN change The user can change his or hers PIN code. PINs can be between 4 and 10 digits.

Reply The user can choose to reply to a received message. The message can be sent as an encrypted or regular SMS.

Delete Messages in the inbox can be deleted.

Frustration The users could use this feature to log their frustation level from 0 through 5, where 5 is the most frustrated.

4.2 Development

The prototype was written in Java2 Micro Edition (J2ME), using the CLDC[1] 1.0 and MIDP[2] 2.0. NetBeans IDE 5.5 with Mobility add-on pack turned out to be a very good tool for the prototype development. Additionally, BouncyCastle [45] provides many cryptographic API's for both Java and .Net(C#), and the lightweight API was perfect for the J2ME prototype. Despite its name, this lightweight API includes numerous acknowledged algorithms like AES, IDEA, SHA-256, and it works great with J2ME. For more info about J2ME, CLDC or MIDP visit SUN Microsystems website [46] or read the Master thesis of Egeberg [47].

[1]Connected Limited Device Configuration
[2]Mobile Information Device Profile

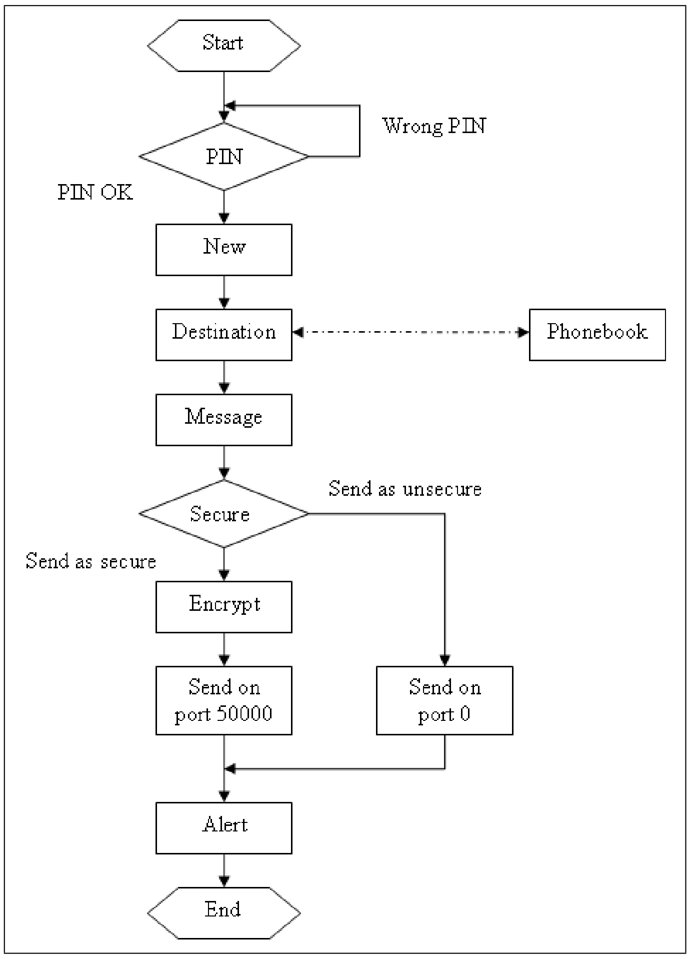


Figure 6: Program flow for sending SMS'es

4.2.1 SMS communication and PushRegistry

J2ME can utilize a set of communication methods, and this is controlled through a Connector-class. Examples of this are SMS, HTTP, HTTPS and Bluetooth. When using the different connections, it is necessary to run them in separate threads, in order to avoid faulty operation (deadlocks e.g.). Appendix D.4 and D.5 shows J2ME code for HTTP-communication and SMS sending, respectively.

One can get a midlet to launch when certain events occur, e.g. receiving an SMS. This technique is called PushRegistry [48]. To get the Midlet to launch when an SMS is received, one must configure this through the Java Application Descriptor (JAD) file, by adding a line, like the following:

```
MIDlet-Push-1: sms://:50000,no.nislab.SMSReceive,*
```

This tells the phone operating system to launch the no.nislab.SMSReceive-midlet when an SMS is received on port 50000.

4.2.2 Midlets

The prototype consisted of 4 midlets:

SecureSMS SecureSMS is the main midlet, containting the functiuonality described in section 4.1.

SecureDecrypt SecureDecrypt is the midlet that runs when an encrypted SMS is received on port 50000. Once received, the midlet then decrypts it and stores the message in the prototype's inbox.

SecureSettings This midlet is used to set the cryptographic keys and to delete the usage log. A secret PIN code prevents the user from running this midlet.

SecureLog This midlet is used to view and export the usage log, by uploading it to a web-server.

4.3 Encrypting and sending an SMS

The content of the SMS messages were encrypted using AES with 192 bits key length. The cipher is run in Cipher Feedback mode (CFB). This makes the block cipher operate like a stream-cipher, and it is very suitable for our purpose, because it doesn't produce much overhead [49]. Figure 6 shows how the programs flow of execution for sending a secure SMS. The J2ME-code for the encryption of an SMS is listed in appendix D.1.

4.4 Limitations

Unfortunately, we didn't have real certificates to sign our midlets with. This meant that the midlets must ask the user for permission to use air-time; that is sending SMS, connecting through GPRS or Bluetooth and so on. This may be viewed upon as an extra obstacle for sending encrypted SMS messages.

5 Experiment

In this chapter we will take a look at the different phases of the experiment, and how they were conducted. The experiment consists of the following parts - introduction, field test and a questionnaire.

5.1 Introduction and training

At the very beginning of the field test, the users got a A5-sized user manual which described three of the features of the prototype;

1. How to send an encrypted SMS
2. How to receive an encrypted SMS
3. Frustration logging

We chose not to educate the users too much, as we wanted to see how well they learned to use the prototype, and figured it would be best if their knowledge about the prototype were at a minimum. The whole manual is reproduced in appendix A.

5.2 Participants

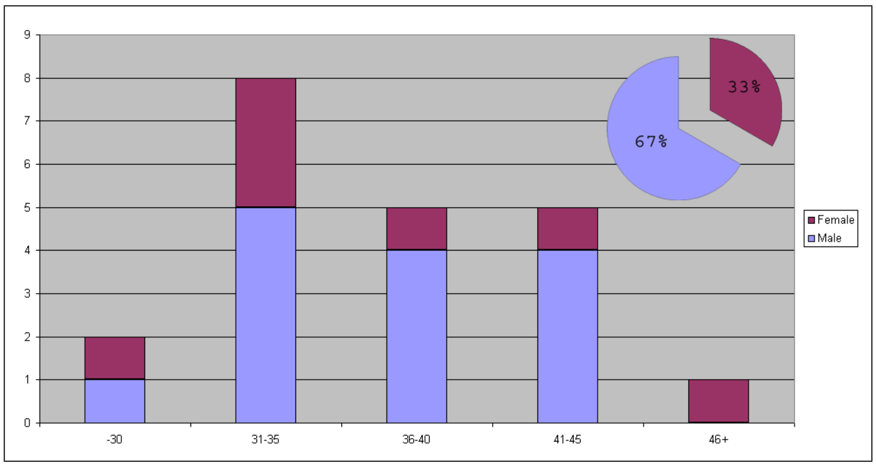


Figure 7: Age and gender distribution

A total of 21 persons participated — 7 females and 14 males — ranging from age 29 to 56. See table 2 for a complete list of the participants. Figure 7 shows the distribution of the participants' age and gender. Their mobile phone experience ranged from 3[1] to 17 years.

[1]We are having a hard time beliving this short mobile phone experience. Maybe the user thought we asked about when he or she got a phone payed by the employer

Participant	Age	Gender	Phone model	Phone experience
P01	32	Male	Nokia 6600	13 years
P02	34	Female	SonyEricsson W800i	12 years
P03	30	Female	SonyEricsson K750i	11 years
P04	29	Male	SonyEricsson K750i	8 years
P05	39	Male	Nokia 6600	10 years
P06	35	Male	SonyEricsson K800i	11 years
P07	56	Female	Nokia 6230i	7 years
P08	37	Male	Nokia 6230i	12 years
P09	33	Male	Nokia 6230i	8 years
P10	36	Male	Nokia 6230i	10 years
P11	44	Male	SonyEricsson K750i	17 years
P12	42	Male	SonyEricsson K750i	13 years
P13	39	Male	Nokia 6230i	13 years
P14	41	Male	SonyEricsson K800i	12 years
P15	41	Male	SonyEricsson K800i	14 years
P16	32	Male	Nokia N70	8 years
P17	34	Female	SonyEricsson K800i	12 years
P18	31	Female	Nokia 6230i	3 years
P19	34	Female	Nokia N70	12 years
P20	40	Male	Nokia N73	8 years
P21	41	Female	SonyEricsson K800i	11 years

Table 2: Participants

5.3 Field test

The participants used their own phone for the field test, and a total of 6 different models were used.

5.3.1 Deployment

The prototype was distributed to the participants with a WAP-push message. A WAP-Push message is a special type of SMS message, instructing the phone to download e.g. ring tones, logos and software. This WAP-push message was sent using a free trial version of NowSMS [50]. The application, consisting of a Java Application Descriptor (JAD) file and Java ARchive (JAR) file, had to be published to a web-server, as the phones download these files through Internet (GPRS, EGDE or UMTS).

The users had to install the application them selfs, which they managed very well. After successfully installing the prototype, we had to manually set the keys on their phones, with the SecureSettings midlet.

5.4 Data capture and gathering

In any Midlet (a J2ME program) the method `commandAction()` is required; because of the `CommandListener` interface the `Midlet` class must implement. The `commandAction()`-method handles all the commands in a midlet, and is a perfect place to "hook" the usage logging system. Every command in the entire midlet is controlled through this single method. The methods `startApp()`, `pauseApp()` and `destroyApp()` are also required in a Midlet, and are also good places to put calls to the log-mechanism.

Furthermore, one would, obviosly, have to put log-mechanism calls in methods and functions of particular interest. In our prototype the method for SMS sending is a good example of this.

Eventually it becomes necessary to move or copy the log from the phone and into a database. The question then becomes; how? On a Java-enabled mobile phone many different API's may be available, such as HTTP-communication, Bluetooth support and so on. Different vendors and phone models supports different API's. The only API one is guaranteed to be supported is the HTTP API. This API was used to export the logs from the phone and into a database. This was done, simply, by looping through the log-records, and making HTTP-calls to a web-server, with the log-entry data as GET-parameters.

Example of the export of a log-entry containing the quit-command of participant P01's phone:

```
http://my-ip/log.aspx?phoneNo=P01&value=2007.04.12:11_10:54_Cmd_quit
```

This call appends the following line to a text file called `P01.txt` on the web-server.

```
2007.04.12 11:10:54 Cmd_quit
```

Finally all the text files were parsed and inserted into a SQL database for easy analysis.

5.5 Questionnaire

We began the design of the questionnaire while the field test took place. The goal was to complement any findings from the usage logs as well as to record demographic data about the participants. The design of the questionnaire was completed only after a preli-menary analysis of the usage logs was done.

6 Results

Whitten and Tygar [2] use a definition for security software and usability, based on 4 principles.
Definition: Security software is usable if the people who are expected to use it:

1. are reliably made aware of the security tasks they need to perform;
2. are able to figure out how to successfully perform those tasks;
3. don't make dangerous errors; and
4. are sufficiently comfortable with the interface to continue using it.

In this thesis we will try to match our results to each of the 4 principles above. To do this we have established a set of indicators, and based on analysis of the usage logs and the answers found in the questionaire, we will see how our indicators fits the 4 principles above.

The first principle can be viewed upon as a premise for the following three. It's a matter of awareness and training in this case. Finally, we will discuss any results that doesn't relate or fit to the described approach above.

6.1 Indicators

In section 3 we created 5 indicators. In this section we will use our usage log data and answers from the questionnaire to calculate the different scores in each indicator.

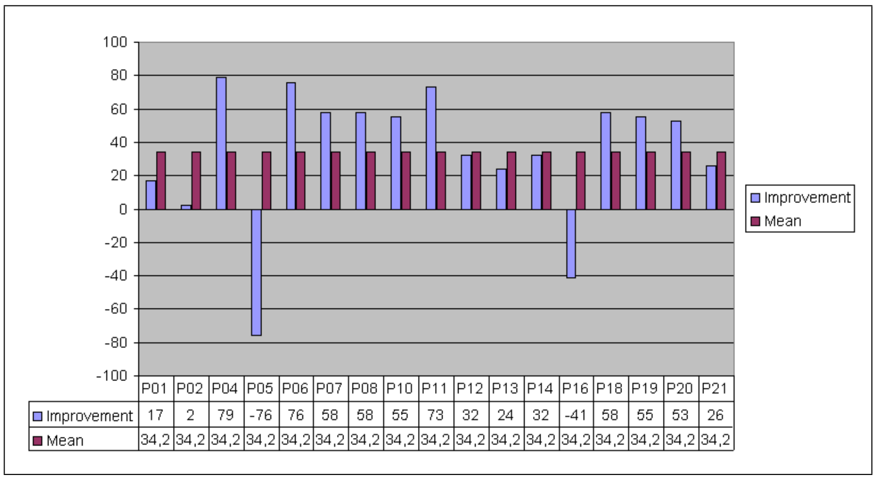
6.1.1 Learning curve (I1)

First, we calculated a trend line for each user, based on the time they spent on sending their messages. Then we used the trend line as a representation of the users' learning curve, and also calculated an improvement value in seconds for each user. See table 3. We chose to do it this way because the standard deviations were too high; that is, the different values recorded fluctuated too much.

Participant	delta	%		Participant	delta	%
P01	2,6	17%		P13	7,0	24%
P02	0,6	2%		P14	7,0	32%
P04	25,0	79%		P16	-9,1	-41%
P05	-9,6	-76%		P18	7,8	58%
P06	25,2	76%		P19	11,0	55%
P07	13,2	58%		P20	14,8	53%
P08	9,3	58%		P21	3,2	26%
P10	14,0	55%		**Mean**	8,4	34%
P11	14,8	73%		**Median**	7,8	53%
P12	6,0	32%		**Std.Dev**	9,6	41%

Table 3: Learning curve

The users had an average decrease in how much time they spent on sending a message by 8 seconds, an average improvement of 34%. Participants P03 and P17 didn't send enough messages to give results on improvement, while P09, and P15 did send messages, but only replies. When you reply to a message the recipients' number is already filled in correctly, and this may skew the comparisons of time spent on each message, hence they are left out of the summary.



	P01	P02	P04	P05	P06	P07	P08	P10	P11	P12	P13	P14	P16	P18	P19	P20	P21
▨ Improvement	17	2	79	-76	76	58	58	55	73	32	24	32	-41	58	55	53	26
▨ Mean	34,2	34,2	34,2	34,2	34,2	34,2	34,2	34,2	34,2	34,2	34,2	34,2	34,2	34,2	34,2	34,2	34,2

Figure 8: Learning curve

As one can read from table 3 and figure 8 there are two participants which stand out from the crowd, P05 and P16, with their lack of improvement. If the users got disturbed while typing a message, this will increase the time spent on a message. It is likely that this is the case for user P16, which used 21, 44, 14, 14 and 17 seconds prior to the last message where he used 50 seconds to complete. See figure 9.

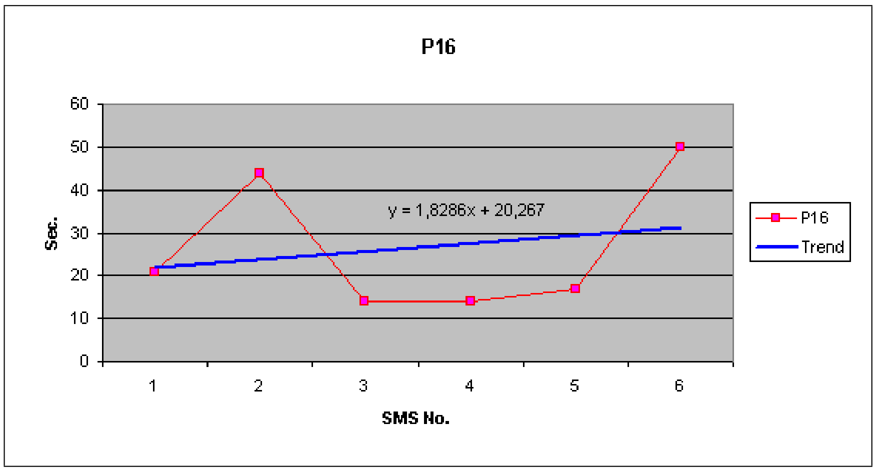


Figure 9: Learning curve for P16

Score #	Value	Importance	Result
Improvement	34%	Crucial	Good

Table 4: Learning Scores

Based on the numbers presented above we can conclude that the vast majority of the users got more efficient and familiar with the prototype.

6.1.2 Awereness (I2)

Based on the answers for the questionnaire, question 1 through 3, one can say that the users do have a genuine need for security, they know that regular SMS isn't particular secure and they make decisions on a daily basis whether the need for information sharing proceeds the need for security. Based on this it is reasonable to say that the users are aware of the different security aspects regarding SMS sending. See table 13, 14 and 15 for details on this.

Sec.mechanism	No. of participants	%
PIN	15	71%
Encryption	15	71%
PIN & Encryption	9	43%

Table 5: Perceived security mechanisms

15 users (71%) mentioned the PIN code, 15 users (71%) mentioned encryption when we asked about which security mechanism the prototype had. Only 9 users (43%) mentioned both PIN and encryption.

$$\text{PIN Score} = \frac{15}{21} = 71\%$$

$$\text{Encryption Score} = \frac{15}{21} = 71\%$$

$$\text{PIN Change Score} = \frac{1}{21} = 5\%$$

Given that 71% think of a PIN code as a security mechanism are good. But, the fact that 71% consider encryption to be a security mechanism in out prototype, are not good enough. The encryption is a major part of the application, and a higher value was expected. Finally, when just 1 user (5%) changes is PIN, we have to rate this as a poor performance.

Score #	Value	Importance	Result
PIN Score	71%	Important	Good
Encryption Score	71%	Crucial	Acceptable
PIN Change Score	5%	Crucial	Poor

Table 6: Awareness Scores

Based on the findings presented above one can conclude that the users scored rather poorly on this indicator.

6.1.3 Frustration levels (I3)

The users didn't use the frustation registration screen very much; only 5 users logged their frustrations. See figure 10.

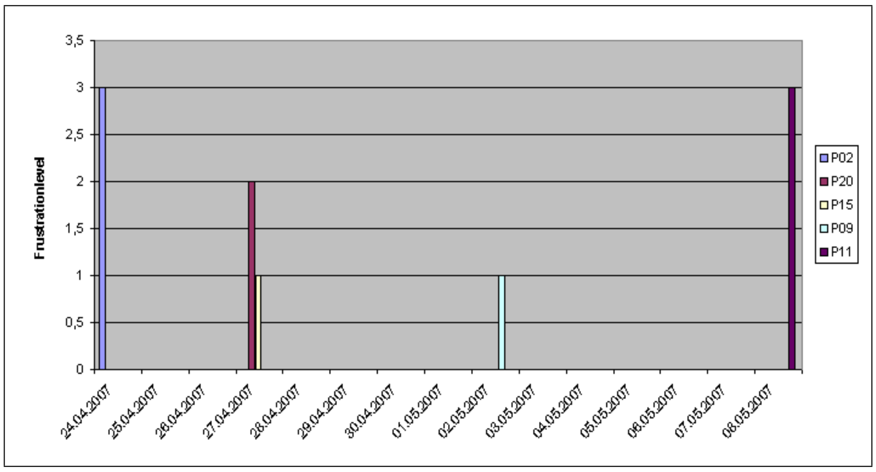


Figure 10: Frustration levels

The fact that only 5 participants used this feature tells us one of three things; firstly the users didn't understand the feature at all. After all, it is an unusual feature. Secondly, the users didn't get very frustrated during their testing. Third, the users got too frustrated to use it; if an application is making you very frustrated, then chances are you won't use any feature it may have, even if it's a frustration feature lik ours.

This indicator could prove useful if the log data had been better. But for this thesis' part we choose not to include this indicator.

6.1.4 Acceptance (I4)

Even though the experiment did last for 32 days, different users did start their tests on different dates. In total the users sent 132 messages. 56 of the messages sent were replies, which mean that 76 messages were composed as new messages.

$$New\ Messages\ Score = \frac{76}{132} = 58\%$$

Feature	No. of users	%
Send encrypted	21	100%
Receive	20	95%
Reply	17	81%
PIN change	1	5%
Delete	10	48%
Send encrypted	21	100%
Mean		66%

Table 7: Features scores

In table 8 one can see the different usage of key features in the prototype. The fatures "Send regular" and "frustration" is left out. Send regular was a feature we didn't want the participants to use, and the frustration level feature was constructed for this thesis, and can not be included as a key feature, from the users' perspective. Based on the data from table 8 we have calculated the features score to 66%. One can easily see that the PIN change usage, of only 5%, lowers the score considerably.

$$Features\ Score = 66\%$$

Score #	Value	Importance	Result
New Messages Score	58%	Important	Good
Features Score	66%	Less important	Acceptable

Table 8: Features Scores

Based on the numbers and figures above we can conclude that the user scores well on this indicator.

6.1.5 Errors (I5)

Of the 132 messages that were sent during the experiment, 4 of them were sent without encryption; as regular plain text messages. We know that 2 of those were intentionally sent just as a test. We had originally planned to treat all plain text messages as errors, but participant P05 told us, unasked, that he had sent 2 messages as insecure as a part of his exploration of the prototype. We choose to trust the participant on this, and exclude those 2 messages from the score calculation.

$$Plain\ Text\ Score = \frac{2}{132} = 1,5\%$$

There were no registered run-time errors in the logs. This means that the prototype was stable and performed well, which also means that bugs and crashes didn't annoy the users. A great deal of effort was put into making the prototype perform flawlessly, and it was comforting to see that the efforts paid off. The users didn't report any crashes either. Table 9 contains the reported issues.

	Date	Issue	Comment
P08	24.04.2007	The phone is acting slower after installation	Nokia 6230i
P08	24.04.2007	Many keystrokes to launch the application	
P17	27.04.2007	Can't open phonebook from application	Configuration issue on SonyEricsson K800i
P12	28.04.2007	My phone doesn't beep when receiving	Configuration issue on Nokia 6230i

Table 9: Issues reported by users

$$\text{Error Score} = (2 \times 10) + 4 = 24$$

Score #	Value	Importance	Result
Plain Text Score	1,5%	Critical	Poor
Error Score	24	Critical	Poor

Table 10: Error Scores

1,5% of the messages were sent as plain text messages. This might not seem to be significantly high. Nevertheless, nearly 2 out of 100 messages are exposed, and given the level of sensitivity these messages may have, it is too high.

Any critical errors are multiplied by 10, to emphasize the importance of not allowing the users to make critical errors. We see that the 2 messages sent as plain text, highly influences both scores. The other 4 issues are completely marginalized. There are some uncertainty attached to the 2 plain text messages; we do not know if they really are errors, or intentionally sent as insecure messages. To be on the safe side we treat them as errors.

Based on this one can say that the majority of the users managed fairly well, while some made critical errors.

6.2 Summary

As mentioned earlier, Whitten and Tygar [2] use 4 principles for usable security software. **Definition:** Security software is usable if the people who are expected to use it:

1. are reliably made aware of the security tasks they need to perform;
2. are able to figure out how to successfully perform those tasks;
3. don't make dangerous errors; and
4. are sufficiently comfortable with the interface to continue using it.

	Poor	**Acceptable**	**Good**
Critical	PIN Change Score (I2) Plain Text Score (I5) Error Score (I5)		Learning Score (I1)
Important		Encryption Score (I2)	PIN Score (I2) New Msg Score (I4)
Less important		Feature Score (I4)	

Table 11: Score and indicator summary

The first one, *security software is usable if the people who are expected to use it are reliably made aware of the security tasks they need to perform*, concerns mostly about proper training, awareness and motivation among the users. This is not something a computer program can achieve alone. In this experiment the users were highly motivated, as they had longed for this type of solution for some while. They were also aware of the different weaknesses and threats in GSM network. But the users didn't get much training. A small manual explaining just the basic bits was all they were given. There was a reason for this; we wanted to see how the users behaved without being particularly skilled or familiar with the prototype.

The indicators show that our results are in compliance with this principle.

The second principle, *security software is usable if the people who are expected to use it are able to figure out how to successfully perform those tasks*, is more complex. First, the learning curve shows that the participants did manage to send encrypted SMS messages, which is the main function of the prototype. They used between 2 to 5 messages to stabilize their time spent at 10-20 seconds when sending messages. The small amount of non-critical errors supports this. But, the users failed considerably regarding their PIN codes. The PIN code was one of the obstacles we choose to have in our prototype, and we deliberately didn't mention anything about how to change the PIN in the manual.

The indicators show that our results is only partially in compliance with this principle.

Principle 3 is *security software is usable if the people who are expected to use it don't make dangerous errors*, and the indicator I5 shows that the users didn't make many errors. But those errors they made are categorized as critical. The only mistakes a user could do were related to the PIN code and the choice of sending messages as "Unsecure". The participants failed regarding the PIN code, and 2 messages was sent as unsecure.

The indicators shows that our prototype is not in compliance with this principle.

The fourth and last principle is *security software is usable if the people who are expected to use it are sufficiently comfortable with the interface to continue using it*. Indicator I1 and I4 shows that the participants did get comfortable with the prototype, and that they are willing to continue to use it. Indicator I3, frustration level, wasn't useful for this thesis, due to lack of data. Nevertheless, the indicator belongs here, and could potentially show that the participants aren't willing to continue to use the application.

The indicators shows that our prototype is in compliance with this principle.

7 Discussion

In this chapter we will discuss our results and the methods we choosed to use. We will also mention some of the experiences we made during this thesis.

7.1 Results

The results show that the users made some errors; didn't change the PIN and sent 2 messages as insecure, regular SMS. When we look at which obstacles we laid out for the users, we see that they "tripped" in them.

The errors they made were possible because we deliberately didn't prevent the users from making them. In a real-life setting the users should be forced to change their PIN, and the prototype should not allow the users to send plain text messages.

We feel it is appropriate to take this into account when we draw our conclusions later on.

7.2 Chosen methods

Now, a few words about how the chosen methods worked out. First of all, a literature study is required when one is doing research. The phrase *standing on the shoulder of giants* illustrates the benefits of a good literature study. The Gjøvik University College library, with their co-operative staff and their scientific databases, provided loads of useful material.

The field-test proved more challenging. Firstly, to get the prototype finished required huge efforts. Mainly because the use of J2ME required us to gain a great deal of new knowledge. And, to get the prototype to perform flawlessly, also required weeks and weeks of programming and testing. It was a huge task to complete. But, the use of a prototype was the only way we could conduct a good field test. The prototype performed well, and gave us lots of useful data.

The questionnaire was not that useful. It did give us a couple of good additional data to back up findings from the field test. But, the 10 questions could easily have been just 5 or 6. Some questions didn't provide enough data for us to conclude. It also required a great deal of labor to get it organized and analyzed.

7.3 Users and security — Past and present

During the last few years we have witnessed a change in how we manage security. In 1999 when Johnny tried, and failed, to use PGP 5.0 to encrypt his emails Johnny (the user) had to manage almost every aspect of his security himself. He had to generate, backup, and publish his keys in the right way. He had to know how these keys worked in order to encrypt, sign or both. Since then we have realized that the users aren't interested in handling this on their own. From the users' perspective, the extra effort required doesn't always exceed the users' perceived need for security. From a business point of view the users should concern themselves with what they are best at, and not spend time fiddling with security issues. Further, we have realized that the default operation always should be the secure option - this has not always been the case.

Security administration has also evolved into a profession, and it is its mission to contribute to make security decisions on behalf of the users. Security software isn't particular secure if the users makes dangerous errors; a high-grade solution may be impossible to break, but it doesn't really matter if the user doesn't use it correct.

However, from the users perspective, the extra effort required doesn't always exceed the users percived need for security.

In 1999, when Johnny tried to encrypt, he used PGP 5.0 - a plug-in to popular email applications like e.g. Eudora. Since then there seems to be consensus amongst the experts that security is not something you bolt on to your application. A plugin to achieve security is exactly an example of this kind of bolt-on solutions. This could be a contributing factor to why Johnny failed to encrypt is emails.

Our prototype was a dedicated application for the purpose of sending secure messages. Maybe dedicated programs is a good approach towards secure and fail-safe security software? One application for regular e-mail and one for secure e-mail. When mixed, the possibilities of errors are evident.

7.4 Bugs and weird behavior

The Nokia 6230i showed some worrying characteristics. One user (P08) thought his phone operated slower after the installation of the prototype. It is hard to tell if this is caused by the prototype alone, but it is likely to believe that the small amount of memory on this phone is the root cause.

Another bug, or feature if you like, emerged when the Nokia 6230i received an encrypted message on port 50000, and launched the receiver midlet. The other phones in the test launched the midlet when a message was received. This midlet then reads the message, decrypts it, stores it in the inbox and displays an alert telling the user that a secure message has been received. But, on the 6230i, the user gets an alert from the phone OS telling him/her about a new message to the prototype. If the user then opens his inbox (the one inside the prototype) the message isn't there. The user actually has to run the receiver-midlet manually in order for the message to be decrypted and stored. This is a major concern, and could possibly cause the users to give up on the system altogether.

One user complained about the lack of alerts (sound and/or vibration) when his Nokia 6230i received messages. This seems to be a configuration-problem on this particular user's phone, as all the other phones did either beep or vibrate, including the other Nokia 6230i in the test.

Finally, one bug did not get solved during the test-period. In J2ME one can tell what type e.g. a textbox is; one can choose from a set of types, where phone number is one of them. This is used in the process of sending a new message. When the user is told to enter the recipients' phone number, this number is typed into a textbox of type phone number. When the KVM1 recognize this, it automatically gives the user the choice of getting this number form the phonebook. This is a nice feature, but on one participant's phone, a SonyEricsson K800i, this didn't work. She was given the choice of opening the phonebook, but when it opened it was empty. She was therefore forced to type in phone numbers manually, and it made her give up on the application. This seems to be a bug on her phone especially.

Another type of textbox is numeric. If one specifies a textbox of this type, the user is only allowed to enter digits between 0 and 9. The textbox for the PIN is of type numeric. On some SonyEricsson K800i phones this made trouble for the users. The default PIN code was "0000". Users with the K800i could not type following zeroes. "00" became just "0", and the users couldn't start the prototype at all. To try to avoid this they tried to enter "10000" and then remove the "1" digit, but then the "0000" left became just "0" again. Hence, the default PIN was changed to "9999".

These errors show that one J2ME application can, and will, behave differently on different phones, and it could potentially result in poor usability.

7.4.1 Usability in J2ME applications - The example of the PIN

The participants had many different phonemodels, and J2ME applications appear and behave differently on them. This is a great challenge for the developer of applications for mobile phones. The simple PIN feature of the prototype is a good example of this. Figure 15, 16, 17 and 18 shows the different actions the user must perform to launch the application. First he or she must choose "Edit", to be able to enter the PIN code into the textbox. When the PIN is entered, he or she must choose "OK", and it brings him/her back to the previous screen, but this time the textbox contains the PIN code. The user must now choose "Options". This opens yet another screen containing "OK" as it's only option. Considering the PIN is correct, choosing "OK" here brings the user to the inbox of the prototype.

On Nokia 6600 all this is done in one screen, and the user can enter digits directly into the textbox. Then he or she chooses "Options" and "OK". See figure 19 and 20. SonyEricsson W800 behaves pretty much like Nokia 6230, with an interesting difference. In step 3 the menu item is called "more" instead of "options". See figure 21. This is something the KVM does, and it is impossible for the system designer to influence this. These types of issues make it very hard to design and build consistent and usable user interfaces. Furthermore, all manuals, help- and training-materials will have to take this fact into account.

Phone	Keystrokes	Screens
Nokia 6230i	4	3
Nokia 6600	2	1
SonyEricsson W800	4	2

Table 12: Keystrokes and screens

However, there are 3rd party solutions available to avoid some of these issues, e.g. mBricks [51]. How well they work, in terms of cross-telephone support, performance and so on, has not been investigated.

Table 12 summarizes the differences on Nokia 6230i, 6600 and SonyEricsson W800.

7.4.2 Usability in J2ME applications - Send Secure or unsecure

Another disturbing difference between the phone models was when the users was given the choice of sending the sms encrypted — "Secure" or as regulars SMS — "Unsecure". After the user has entered the recipients phonenumber, the user is faced between two options; "Secure" or "Unsecure". The exact java-code is listed in appendix D.6. The screen consists of only two commands, one for "Secure" and "Unsecure" - there is no back or cancel buttons. On SonyEricsson W800 presents this screen very well, see figure 22.

Nokia 6230i presents this screen totally different. See figure 23 and 24. Here the "Secure"-option is placed an the action-button (push the joystick down), the right softmenu-itme is empty and a KVM-generated item "Options" is placed at the left. The "Unsecure"-option is placed in this menu. Lucky for us, the "Sceure"-option was placed easy accessible at the action-button, but the two buttons are programmed exactly the same, and it could just as easily been the other way around. If so, we belive the number of errors (messages sent as regular SMS) would be higher. The performance and satisfaction level would probably drop also, as the secure option then would require 2 extra keystrokes.

8 Conclusions

This thesis set out to find out if Whitten and Tygar's [2] findings regarding the usability of PGP on personal computers would be the same in a mobile setting. Their work was presented in 1999, and since then we communicate more and more through SMS messages. In section 1.5 we formulated the research questions, and we will use them here, to draw conclusions from our efforts in this thesis.

Which solutions available today offer secure messaging on mobile phones, and what are their costs, pros and cons?

This thesis has shown that many solutions exist, and that there are, from an ordinary person's perspective, considerable drawbacks attached to many of them. Section 2.1 summarizes what we found during our study. Generally, one can say they are expensive and aimed towards government customers. Further, most of the vendors don't share important data about their products, making it hard to do a thorough data collecting.

Section 2 sums up both existing solutions and relevant research on this topic, even if it is not complete.

Even with the goal of making security transparent in the software, the user will have to perform a minimum of security related tasks. How well will Johnny perform those tasks?

We made a secure messaging prototype for mobile phones. The prototype offered security look'n'feel and encryption of the SMS content. But, we also introduced two obstacles; we did'nt tell or force Johnny to change the PIN code and we gave him the opportunity to make a dangerous error each time he sent an SMS message. He could choose to send his messages unsecurely, which is a dangerous thing to do in a prototype especially built for secure messaging.

- Will he get comfortable with the software?

Johnny did get comfortable with the software. Despite the almost total absence of training and help, Johnny was able to install and operate the prototype very well. He got familiar with all the different features of the prototype, except the PIN change and frustration level screens.

Johnny got comfortable with the software.

- How fast will he learn the software?

Johnny tested the software for just a few days, and during those days he learned to send messages over 30% faster. Naturally, Johnny needed some time to get acquainted with how the prototype worked. He also had to learn how his phone behaved, as different phone models handles J2ME programs differently. This wasn't something he could learn from a manual. And chances were that his friends had another phone model, so he couldn't ask them either.

Johnny learned the software fast.

- Will he make errors?

First of all, Johnny doesn't think of the PIN as a mechanism for security. This is probably the reason why he didn't change the PIN. Secondly, Johnny made errors when he sent messages too, and even if the amounts are very small, we still have to call them dangerous.

Johnny did make a small amount of dangerous errors. But, these errors were only allowed because we deliberately let them to. It is fair to say that Johnny did manage without making dangerous errors.

Additional findings

As shown in section 7.4.1 and 7.4.2 we have shown that good usability with standard J2ME is a major challenge. We found it hard to keep track of all the different behaviors on the phone models we were developing for. If one are making software for the large market (nation-wide or bigger), one will be faced with enormous challenges regarding consistent and user-friendly user interfaces.

9 Future work

The secure messaging term includes many techniques, such as encryption, signing, non-repudiation. Before a prototype can offer true secure messaging on a mobile phone, better and more security has to build into the application. Our prototype offered mostly security look'n'feel, as we concentrated on the usability aspect. It would be interesting to see how a better, that is, more in compliance with the terminology, secure messaging scheme proposed for a mobile phone. Further, we conlcuded that building applications with good usability with standard J2ME is diffucult. There are ways to mitigate this, such as mBricks [51]. A research investigating what effect and impact such solutions have compared to standard J2ME in terms of usability and effeciency could prove very useful.

Mobile phones have good connectivity; they can communicate through speech and SMS (GSM) and over the Internet (GPRS/EDGE/UMTS). But, still it is both expensive and power-consuming to use extensively. With ADSL and cable modem connections a pc can stay online 24/7 and have great bandwidth available. A mobile phone doesn't have these opportunities, and the need of efficient network prototcols is crucial. To be able to have a better and more secure messaging system, one would need to use key exchange protocols. But, to implement those with the limited bandwidth available on a mobile phone may not be a good idea. Therefore, research about how to implement efficient and safe key exchange protocols for mobile phones could also bring the security on mobile phones one step ahead. Oppliger [17] speaks of *simultaneous secret exchange*, and this method should be feasible to implement on small devices.

Finally, usability related research often uses participants in different types of tests. We found the numbers of participants in such types of tests low. Landauer and Nielsen [44] present an approximation formula to help determine how many participants a usability test would require. This formula requires a p-value - the probability that any single usability issue would be found by any single participant. That p-value is hard to estimate, especially when testing usability on mobile phones. Research on how to determine a good p-value is therefore needed.

Bibliography

[1] R.Safavi-Naini, W.Susilo, G. 2001. Towards securing 3G mobile phones (extended abstract). In *Networks, 2001. In Proceedings Ninth IEEE International Conference*, 222–227. IEEE.

[2] Alma Whitten, J. T. 1999. Why johnny can't encrypt: A usability evaluation of PGP 5.0. In *Proceedings of the 8th USENIX Security Symposium*, 169–184. USENIX.

[3] Post og Teletilsynet. 2006. Norwegian post and telecommunacation authority - statistics 2006. http://www.npt.no/. (Last visited May. 2007).

[4] SkandiaBanken. 2005. Skandiabanken. http://www.skandiabanken.no. (Last visited Nov. 2005).

[5] 2001. Datatilsynet krever krypteringsstyrke tilsvarende DES128. http://www.datatilsynet.no/templates/article____889.aspx. (Last visited May. 2007).

[6] Rubin, J. 1994. *Handbook of usability testing: how to plan, design and conduct effective tests*. John Wiley & Sons, Inc.

[7] Rohde & Schwarz. 2005. Test and measurement and communication systems by Rohde & Schwarz (homepage). http://www.rohde-schwarz.com/. (Last visited Nov. 2005).

[8] Forsvarsnett www.mil.no. 2002. Ny kryptotelefon NSK 200 er produsert i norge, press release 2002-03-06. http://www.mil.no/start/aktuelt/pressemeldinger/article.jhtml?articleID=12352. (Last visited Feb. 2007).

[9] 2005. Global-teck worldwide homepage. http://www.global-teck.com/english/telecomproducts.php. (Last visited Dec. 2005).

[10] Schneier, B. 1999. Cryptography: The importance of Not being different. In *Computer*, volume 32, 108–109, 112. IEEE Computer Society.

[11] Common Criteria. 2007. Common Criteria portal. http://www.commoncriteriaportal.org/. (Last visited May. 2007).

[12] NIST. 2007. National Institute of Standards and Technology. http://www.nist.gov/. (Last visited May. 2007).

[13] NSM. 2007. Norwegian National Security Authority. http://www.nsm.stat.no/. (Last visited May. 2007).

[14] Microsoft. 2007. Windows mobile direct push email. http://www.microsoft.com/windowsmobile/business/directpushemail.mspx. (Last visited May. 2007).

[15] Nødnett. 2007. The directorate for emergency communication homepage. http://www.dinkom.no/. (Last visited Jun. 2007).

[16] TETRA. 2007. Terrestrial trunked radio (TETRA) homepage. http://www.tetramou.com/default.aspx. (Last visited Jun. 2007).

[17] Oppliger, R. 2004. Certified mail: The next challenge for secure messaging. *Communication of the ACM*, 47(8), 75–79.

[18] Volker Roth, Tobias Strauß, K. R. 2005. Security and usability engineering with particular attention to electronic mail. In *International Journal of Human-Computer Studies*, volume 63, 51–73. Elsevier.

[19] Rebecca E. Grinter, D. K. S. 2003. Three challenges for embedding security into applications. *Proceedings of CHI 2003 Workshop on HCI and Security Systems*.

[20] John Viega, G. M. 2001. *Building Secure Software*. Addison-Wesley.

[21] Yee, K.-P. 2004. Aligning security and usability. In *Security Privacy Magazine, IEEE*, volume 2, 48–55. IEEE.

[22] Anand Raghunathan, Srivaths Ravi, S. H. . J.-J. Q. 2003. Securing mobile appliances: New challenges for the system designer. *Design, Automation and Test in Europe Conference and Exhibition 2003, DATE'03*, 176–181.

[23] Lindholm, C., Keinonen, T., & Kiljander, H. 2003. *Mobile usability : how Nokia changed the face of the mobile phone*. McGraw-Hill.

[24] Maguire, M. 2004. Does usability = attractiveness?). *Design and emotion : the experience of everyday things - edited by Deana McDonagh et al.*, 303–307.

[25] Vicente, K. 2003. *The human factor : revolutionizing the way people live with technology*. Routledge.

[26] Soriano, C., Raikundalia, G. K., & Szajman, J. 2005. A usability study of short message service on middle-aged users. In *OZCHI '05: Proceedings of the 19th conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction*, 1–4, Narrabundah, Australia, Australia. Computer-Human Interaction Special Interest Group (CHISIG) of Australia.

[27] Nielsen, J. & Mack, R. L. 1994. *Usability inspection methods*. John Wiley & Sons, Inc.

[28] Lewis, C., Polson, P. G., Wharton, C., & Rieman, J. 1990. Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces. In *CHI '90: Proceedings of the SIGCHI conference on Human factors in computing systems*, 235–242. ACM Press.

[29] Jakob Nielsen, R. M. 1990. Heuristic evaluation of user interfaces. In *CHI '90: Proceedings of the SIGCHI conference on Human factors in computing systems*, 249–256. ACM Press.

[30] Lim, Y.-K., Pangam, A., Periyasami, S., & Aneja, S. 2006. Comparative analysis of high- and low-fidelity prototypes for more valid usability evaluations of mobile devices. In *NordiCHI '06: Proceedings of the 4th Nordic conference on Human-computer interaction*, 291–300. ACM Press.

[31] Henry Been-Lirn Duh, Gerald C. B. Tan, V. H.-h. C. 2006. Usability evaluation for mobile device: a comparison of laboratory and field tests. In *MobileHCI '06: Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, 181–186. ACM Press.

[32] McGraw, G. & Felten, E. 1999. *Securing Java: getting down to business with mobile code*. Wiley.

[33] Stone, D., Jarrett, C., Woodroffe, M., & Minocha, S. 2005. *User interface design and evaluation*. Elsevier.

[34] Smetters, D. K. & Grinter, R. E. 2002. Moving from the design of usable security technologies to the design of useful secure applications. In *NSPW '02: Proceedings of the 2002 workshop on New security paradigms*, 82–89, New York, NY, USA. ACM Press.

[35] Microsoft TechNet. 2000. 10 immutable laws of security administration. http://www.microsoft.com/technet/archive/community/columns/security/essays/10salaws.mspx. (Visited Jun. 2007).

[36] Cranor, L. F. & Garfinkel, S. 2005. *Security And Usability: Designing Secure Systems That People Can Use*. O'Reilly.

[37] Michael C. Howard, David LeBlanc, J. V. 2005. *19 deadly sins of software security : programming flaws and how to fix them*. McGraw-Hill/Osborne.

[38] Clare-Marie Karat, John Karat, C. B. 2005. Why HCI research in privacy and security is critical now. In *International Journal of Human-Computer Studies*, volume 63, 1–4. Elsevier.

[39] Nemeth, C. P. 2004. *Human factors methods for design : making systems human-centered*. CRC Press.

[40] Samuel G. Charlton, T. G. O. 2002. *Handbook of human factors testing and evaluation*. Mahwah, N.J. : Erlbaum.

[41] Nielsen, J. 1993. *Usability Engineering*. Academic Press.

[42] IBM. 2002. J2ME record management store. http://www.ibm.com/developerworks/library/wi-rms/. (Last visited Jun. 2007).

[43] Travis, D. 2003. *E-commerce usability*. Taylor & Francis.

[44] Jakob Nielsen, T. K. L. 1993. A mathematical model of the finding of usability problems. In *CHI '93: Proceedings of the SIGCHI conference on Human factors in computing systems*, 206–213, New York, NY, USA. ACM Press.

[45] Bouncy Castle. 2007. The legion of the bouncy castle. http://www.bouncycastle.org/. (Last visited May. 2007).

[46] SUN Microsystems. 2007. The Java ME platform. http://java.sun.com/javame/index.jsp. (Last visited May. 2007).

[47] Egeberg, T. 2006. Storage of sensetive data in a java enabled cell phone. *Master thesis - NISLAB, Gjøvik University College*, 7–9.

[48] SUN Microsystems. 2003. The MIDP 2.0 Push Registry. http://developers.sun.com/techtopics/mobility/midp/articles/pushreg/. (Last visited Jun. 2007).

[49] Stallings, W. 2003. *Network Security Essentials — Applications and standards*. Prentice Hall.

[50] Now SMS. 2007. Reading from the web. http://www.nowsms.com. (Visited May 2007).

[51] mBricks. 2006. mBricks Mobile Application Development Kit. http://www.mbricks.no/. (Last visited Jun. 2007).

A User manual

Hvordan sende en kryptert SMS

- Start programmet `SecureSMS`
- Skriv inn PIN-kode (Default 9999)
- Velg `New`
- Skriv mottagers telefonnumme (Her tilbyr de fleste telefoner å hente nr fra kontaktlista)
- Velg `Next`
- Velg `Secure` (`Unsecure` sender vanlig SMS)
- Skriv meldingen
- Velg `Send`

Hvordan motta en kryptert SMS

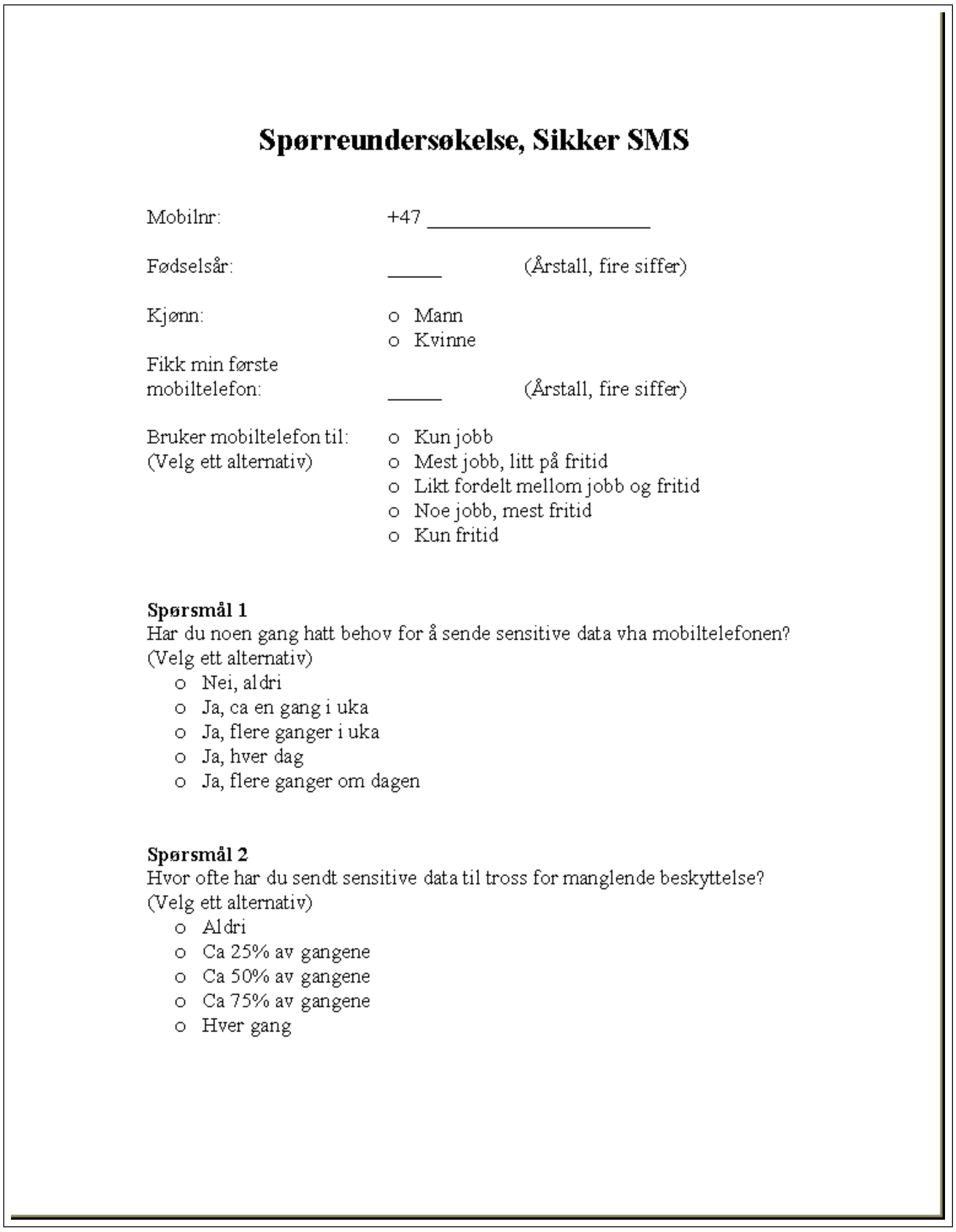
- Krypterte SMS'er sendt til din telefon, mottas av `SecureDecrypt` automatisk (Noen telefoner krever likevel at du kjører programmet manuelt)
- Meldingen dekrypteres nå, og legges i innboksen til `SecureSMS`
- En meldingsboks/skjerm forteller at en kryptert SMS er mottatt
- Lukk meldingsboksen
- Start `SecureSMS`, og meldingen skal ligge i startskjermbildet/innboksen

Frustrasjonsnivå Det er et valg, `Frustration`, hvor du som bruker kan angi på en skala fra 0 - 5 hvor frustrert du er over programmet, dets virkemåte og funksjoner.

- Start programmet `SecureSend`
- Skriv inn PIN-kode
- Velg `Frustration`
- Angi med joystick eller piltaster en verdi - et frustrasjonsnivå
- Velg `Set`. Verdien lagres og du ledes tilbake til innboksen

Frustrasjonsnivået lagres, og er nyttig info til meg om hvordan dere opplever bruken av applikasjonen. Programmene `SecureLog` og `SecureSettings` skal dere foreløpig ikke tukle med.

B Questionaire



Spørreundersøkelse, Sikker SMS

Mobilnr:	+47 ______________________	
Fødselsår:	_____	(Årstall, fire siffer)
Kjønn:	o Mann o Kvinne	
Fikk min første mobiltelefon:	_____	(Årstall, fire siffer)
Bruker mobiltelefon til: (Velg ett alternativ)	o Kun jobb o Mest jobb, litt på fritid o Likt fordelt mellom jobb og fritid o Noe jobb, mest fritid o Kun fritid	

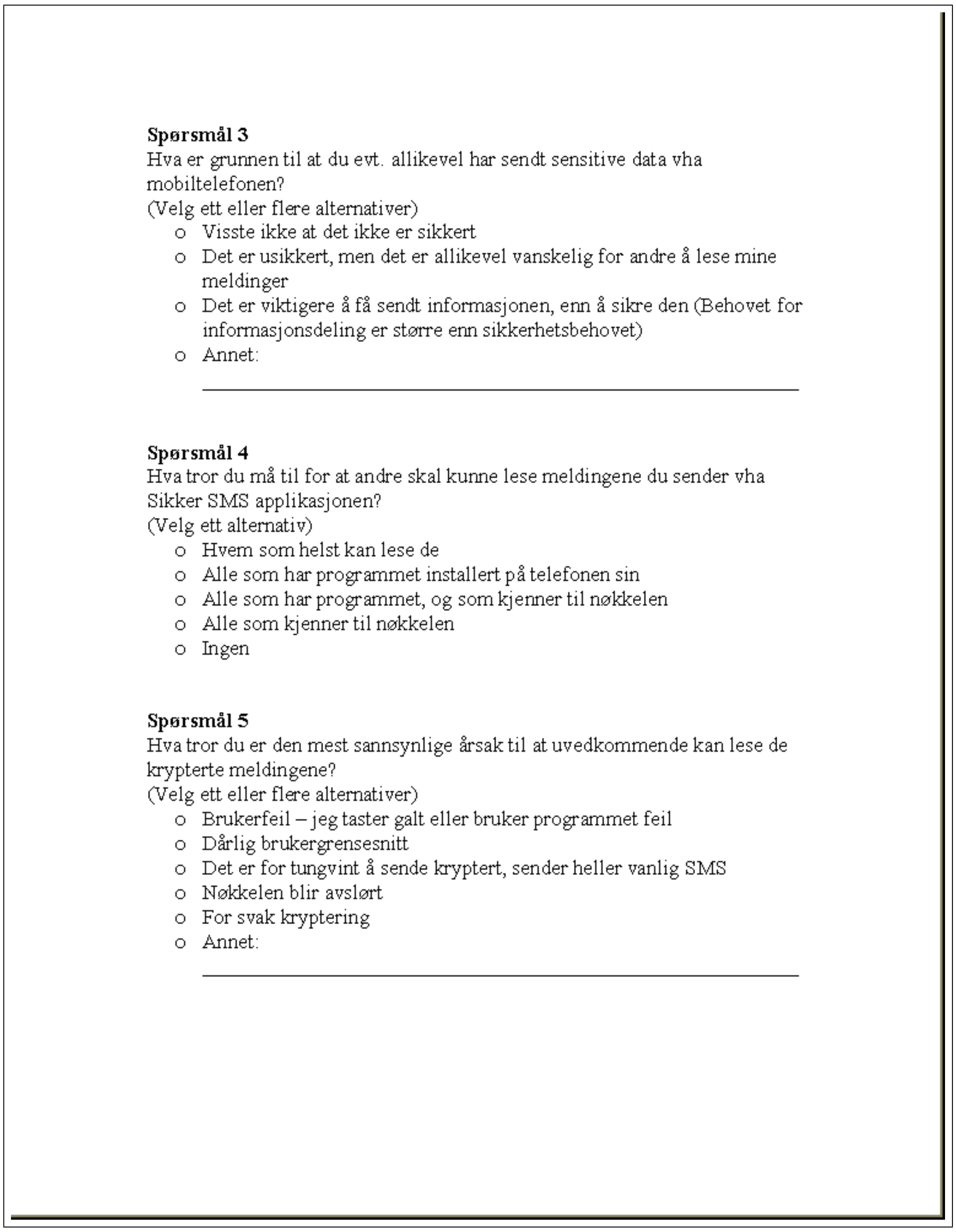
Spørsmål 1
Har du noen gang hatt behov for å sende sensitive data vha mobiltelefonen?
(Velg ett alternativ)

- o Nei, aldri
- o Ja, ca en gang i uka
- o Ja, flere ganger i uka
- o Ja, hver dag
- o Ja, flere ganger om dagen

Spørsmål 2
Hvor ofte har du sendt sensitive data til tross for manglende beskyttelse?
(Velg ett alternativ)

- o Aldri
- o Ca 25% av gangene
- o Ca 50% av gangene
- o Ca 75% av gangene
- o Hver gang

Figure 11: Questionnaire, page 1



Spørsmål 3
Hva er grunnen til at du evt. allikevel har sendt sensitive data vha mobiltelefonen?
(Velg ett eller flere alternativer)

- o Visste ikke at det ikke er sikkert
- o Det er usikkert, men det er allikevel vanskelig for andre å lese mine meldinger
- o Det er viktigere å få sendt informasjonen, enn å sikre den (Behovet for informasjonsdeling er større enn sikkerhetsbehovet)
- o Annet: ______________________________

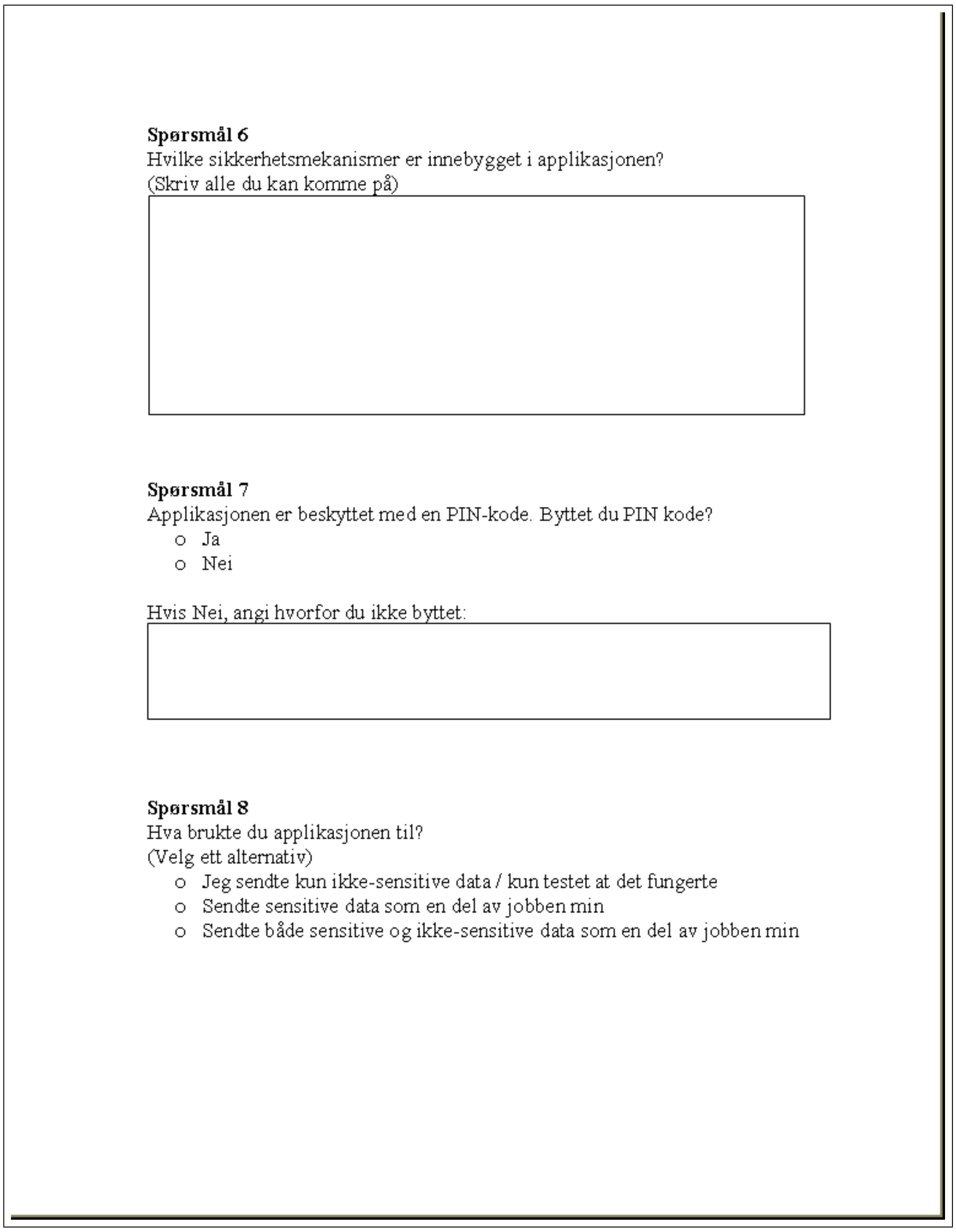
Spørsmål 4
Hva tror du må til for at andre skal kunne lese meldingene du sender vha Sikker SMS applikasjonen?
(Velg ett alternativ)

- o Hvem som helst kan lese de
- o Alle som har programmet installert på telefonen sin
- o Alle som har programmet, og som kjenner til nøkkelen
- o Alle som kjenner til nøkkelen
- o Ingen

Spørsmål 5
Hva tror du er den mest sannsynlige årsak til at uvedkommende kan lese de krypterte meldingene?
(Velg ett eller flere alternativer)

- o Brukerfeil – jeg taster galt eller bruker programmet feil
- o Dårlig brukergrensesnitt
- o Det er for tungvint å sende kryptert, sender heller vanlig SMS
- o Nøkkelen blir avslørt
- o For svak kryptering
- o Annet: ______________________________

Figure 12: Questionnaire, page 2



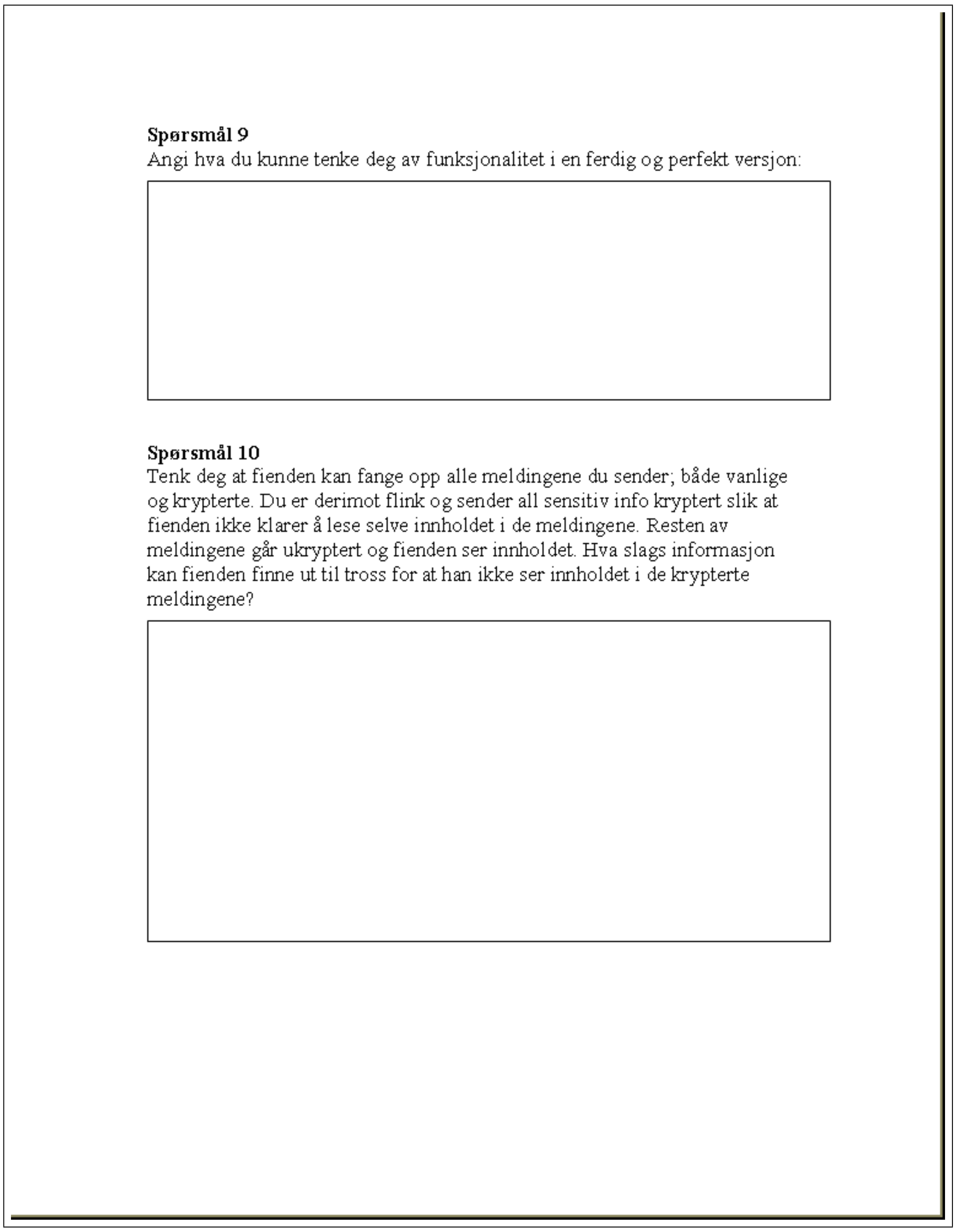
Spørsmål 6
Hvilke sikkerhetsmekanismer er innebygget i applikasjonen?
(Skriv alle du kan komme på)

Spørsmål 7
Applikasjonen er beskyttet med en PIN-kode. Byttet du PIN kode?
- o Ja
- o Nei

Hvis Nei, angi hvorfor du ikke byttet:

Spørsmål 8
Hva brukte du applikasjonen til?
(Velg ett alternativ)
- o Jeg sendte kun ikke-sensitive data / kun testet at det fungerte
- o Sendte sensitive data som en del av jobben min
- o Sendte både sensitive og ikke-sensitive data som en del av jobben min

Figure 13: Questionnaire, page 3



Spørsmål 9
Angi hva du kunne tenke deg av funksjonalitet i en ferdig og perfekt versjon:

Spørsmål 10
Tenk deg at fienden kan fange opp alle meldingene du sender; både vanlige og krypterte. Du er derimot flink og sender all sensitiv info kryptert slik at fienden ikke klarer å lese selve innholdet i de meldingene. Resten av meldingene går ukryptert og fienden ser innholdet. Hva slags informasjon kan fienden finne ut til tross for at han ikke ser innholdet i de krypterte meldingene?

Figure 14: Questionnaire, page 4

C Answers from questionaire

Here we present the answers from the questionnaire.

Question 1 - Have you ever had the need to send sensitive data with your mobile phone?

	Question 1	#	%
A	No, never	0	0%
B	Yes, once a week	12	57%
C	Yes, several times a week	6	29%
D	Yes, every day	2	9%
E	Yes, several times a day	1	5%

Table 13: Question 1

Question 2 - How often have sent sensitive data despite the lack of security?

	Question 2	#	%
A	Never	3	14%
B	Yes, 25% of the times	16	76%
C	Yes, 50% of the times	2	9%
D	Yes, 75% of the times	0	0%
E	Every time	0	0%

Table 14: Question 2

Question 3 - What's the main reason why you still have chosen to send sensitive data with your mobile phone?

	Question 3	#	%
A	Didn't know it was unsecure	0	0%
B	It's still hard for others to read my SMS's	4	19%
C	The need to share info proceeds the need for protection	17	81%
D	Other	0	0%

Table 15: Question 3

Question 4 - What is required for others to be able to read the messages you send with the prototype?

	Question 4	#	%
A	Anyone	0	0%
B	Those with program	2	9%
C	Program and key	13	62%
D	Those who know the key	6	29%
E	Noone	0	0%

Table 16: Question 4

Question 5 - What is the most likely reason that others can read the messages you send with the prototype?

	Question 5	#	%
A	User error	8	38%
B	Poor usability	0	0%
C	Too bothersome to use prototype	9	43%
D	The key is revealed	5	24%
E	Too weak encryption	7	33%
F	Other, please specify	0	0%

Table 17: Question 5

Question 6 - Which security mechanisms were built into the application?

	Question 6	#	%
	Mentioned PIN	15	71%
	Mentioned encryption	15	71%
	Mentioned placing of application	1	5%

Table 18: Question 6

Question 7 - The application was protected with a PIN code. Did you change it? If No, state a reason for this.

	Question 7	#	%
	No, I did not change the PIN	20	95%
	Yes, I changed the PIN	1	5%

Table 19: Question 7

Why didn't you change the PIN	#	%
It was only a test	7	33%
Didn't know it was possible	5	24%
Lazy me! I'll get better	7	33%
It didn't work	1	5%
Other reasons	1	5%

Table 20: Why didn't you change PIN

Question 8 - What did you use the application to?

	Question 8	#	%
	I sent mostly non-sensitive data / only tested the application	11	52%
	I sent sensitive data as part of my job	0	0%
	I sent both non- and sensitive data as part of my job	10	48%

Table 21: Question 8

Question 9 - Name the functionality you would like to have in a perfect secure messaging application?

	Question 9	#	%
	Behave like normal SMS application	6	24%
	Use phonebook when receiving also	12	57%
	Ability to send pictures	15	71%
	Messages to many recipients / group messages	8	38%
	Store messages encrypted on phone	1	5%
	Shortcut to application	1	5%

Table 22: Question 9

Question 10 - Imagine that an adversary could capture the messages you sent, both the regular and the encrypted. However, the the adversary is not able to read the content of the encrypted messages. What could the adversary still possibly find out based on the encrypted and regular messages you send?

	Question 10	#	%
	Trafic analysis	8	38%
	Position / localize my phone	6	24%
	Survey my circle of acquaintances / friends	6	24%
	My name	1	5%
	My type of phone and SIM card	1	5%

Table 23: Question 10

D J2ME source code

D.1 Encryption

```
private byte[] encrypt(String plain) {
  byte[] ciphertext;
  try {
    // Create cipher
    AESLightEngine blockCipher = new AESLightEngine();
    CFBBlockCipher cfbCipher   = new CFBBlockCipher(blockCipher,8);
    StreamCipher streamCipher  = new StreamBlockCipher(cfbCipher);

    // Get key from RMS store
    byte[] keyBytes = getKey();
    KeyParameter key = new KeyParameter(keyBytes);

    // Create IV from random data.
    byte[] iv = new byte[blockCipher.getBlockSize()];
    SecureRandom mSecureRandom = new SecureRandom();
    mSecureRandom.nextBytes(iv);

    // Concatenate IV and the message.
    byte[] messageBytes = plain.getBytes("UTF-8");
    int length          = messageBytes.length + blockCipher.getBlockSize();
    byte[] plaintext    = new byte[length];
    System.arraycopy(iv, 0, plaintext, 0, iv.length);
    System.arraycopy(messageBytes, 0, plaintext, iv.length, messageBytes.length);

    // Encrypt the plaintext.
    ciphertext   = new byte[plaintext.length];
    streamCipher.init(true, key);
    streamCipher.processBytes(plaintext, 0, plaintext.length, ciphertext, 0);
    return ciphertext;
  } catch(Exception ex) {
    System.out.println(ex.getMessage());
    return null;
  }
}
```

D.2 getKey()

```
private byte[] getKey() {
  RecordEnumeration records = null;
  RecordStore settings = null;
  int id;
  byte[] data;
  String value;
  byte[] ret = null;
  try {
    settings = RecordStore.openRecordStore("secSettings", true);
    records = settings.enumerateRecords(null, null, false);
    while (records.hasNextElement()) {
      id = records.nextRecordId();
      data = settings.getRecord(id);
      byte[] key = new byte[4];
      key[0] = data[0];
      key[1] = data[1];
      key[2] = data[2];
      key[3] = data[3];
      value = new String(key);
      if (value.startsWith("key;")) {
        ret = Util.getBytes(data, 4);
      }
    }
    settings.closeRecordStore();
  } catch(Exception ex) {
    Util.logThis("getKey - " + ex.message);
    return null;
  }
  return ret;
}
```

D.3 setKey()

```
private void setKey(String newKey) {
  RecordEnumeration records = null;
  RecordStore settings = null;
  int toDelete = -1;
  try {
    settings = RecordStore.openRecordStore("secSettings", true);
    records = settings.enumerateRecords(null, null, false);
    int id;
    byte[] data;
    String value;
    while (records.hasNextElement()) {
      id = records.nextRecordId();
      data = settings.getRecord(id);
      byte[] key = new byte[4];
      key[0] = data[0];
      key[1] = data[1];
      key[2] = data[2];
      key[3] = data[3];
      value = new String(key);
      if (value.startsWith("key;")) {
        toDelete = id;
        settings.deleteRecord(toDelete);
      }
    }
    byte[] hashBytes = Util.hashThis(newKey);
    byte[] keyBytes = "key;".getBytes();
    byte[] newData = concatBytes(keyBytes, hashBytes);
    settings.addRecord(newData, 0, newData.length);
    settings.closeRecordStore();
  } catch(Exception ex) {
    Util.logThis("setKey - " + ex.message);
  }
}
```

D.4 Communicate with HTTP

Below is an example of how to communiacte with HTTP:

```
StreamConnection conn = null;
InputStream in = null;
StringBuffer webRes = new StringBuffer();
String url = "http://www.foo.bar";
try {
  conn = (StreamConnection) Connector.open(url);
  in   = conn.openInputStream();
  int ch;
  while ( ( ch = in.read() ) != -1 ) {
    webRes.append((char) ch);
  }
  in.close();
  conn.close();
  in = null;
  conn = null;
} catch(Exception ex) {
  Util.logThis(ex.message);
  webRes = ex.getMessage();
}
```

D.5 Sending of SMS

This is an example of how to send a regular SMS:

```
String address = "+4712345678";
String message = "This is the textmessage";
MessageConnection smsconn = null;
try {
  // Regular SMS uses port 0, hence the :0 appended to the address
  address = address + ":0";
  smsconn = (MessageConnection)Connector.open(address);
  TextMessage txtmessage = (TextMessage)
    smsconn.newMessage(MessageConnection.TEXT_MESSAGE);
  txtmessage.setAddress(address);
  txtmessage.setPayloadText(message);
  smsconn.send(txtmessage);
  smsconn.close();
} catch (Throwable t) {
  //
}
```

Next is an example of how to send a encrypted SMS:

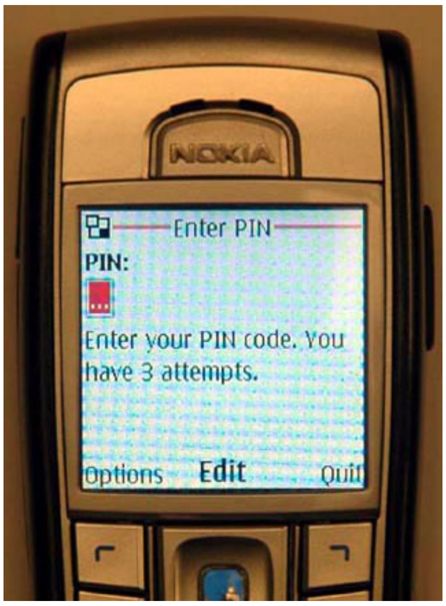
```
String address = "+4712345678";
String message = "";
String smsPort = "50000";
MessageConnection smsconn = null;
try {
  // Encrypted messages uses port 50000, hence the :50000
  address = address + ":" + smsPort;
  smsconn = (MessageConnection)Connector.open(address);
  // encrypted messages are binary messages
  BinaryMessage binmessage = (BinaryMessage)
  smsconn.newMessage(MessageConnection.BINARY_MESSAGE);
  binmessage.setAddress(address);
  // The encrypt()-function accepts a string,
  // and returns a byte-array
  byte[] cipherMessage = encrypt(message);
  binmessage.setPayloadData(cipherMessage);
  smsconn.send(binmessage);
  smsconn.close();
} catch (Throwable t) {
  //
}
```

D.6 Send as Secure or Unsecure

This is the code used to generete the screen where the user can choose to encrypt the message:

```
secureCommand = new Command("Secure", Command.ITEM, 1);
unsecureCommand = new Command("Unsecure", Command.ITEM, 1);
secureUnsecureAlert = new Alert("Secure or unsecure?");
secureUnsecureAlert.setString("Choose secure to encrypt " +
                              "the SMS, or unsecure to send " +
                              "as plain SMS");
secureUnsecureAlert.setType(AlertType.CONFIRMATION);
secureUnsecureAlert.setTimeout(Alert.FOREVER);
secureUnsecureAlert.addCommand(secureCommand);
secureUnsecureAlert.addCommand(unsecureCommand);
secureUnsecureAlert.setCommandListener(this);
```

E Screen shots



Enter your PIN code. You have 3 attempts.

Figure 15: Nokia 6230i - PIN screen 1

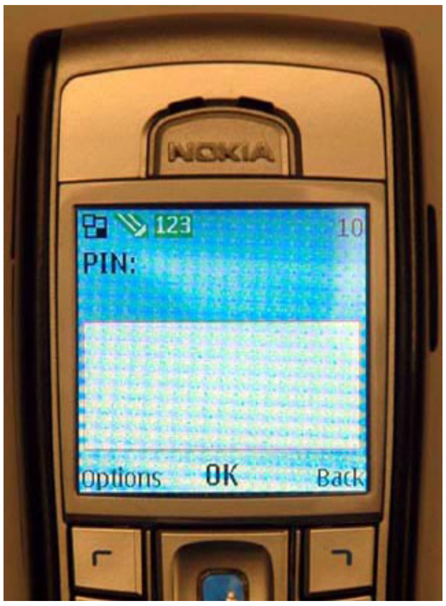


Figure 16: Nokia 6230i - PIN screen 2

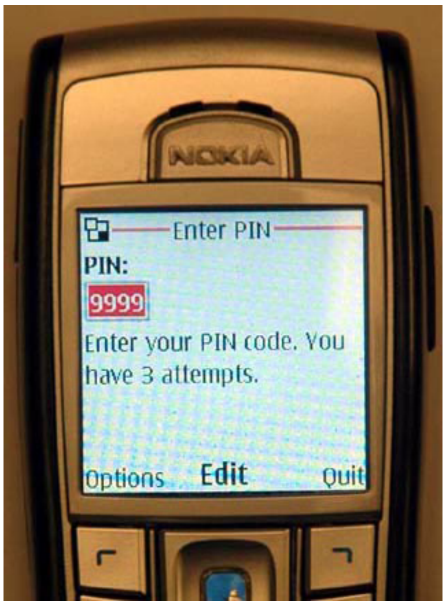


Figure 17: Nokia 6230i - PIN screen 3

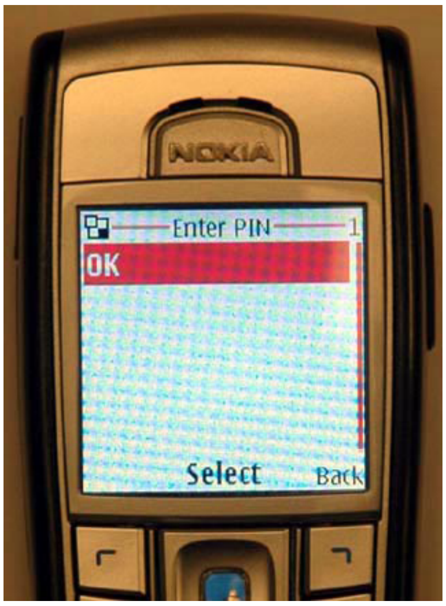


Figure 18: Nokia 6230i - PIN screen 4

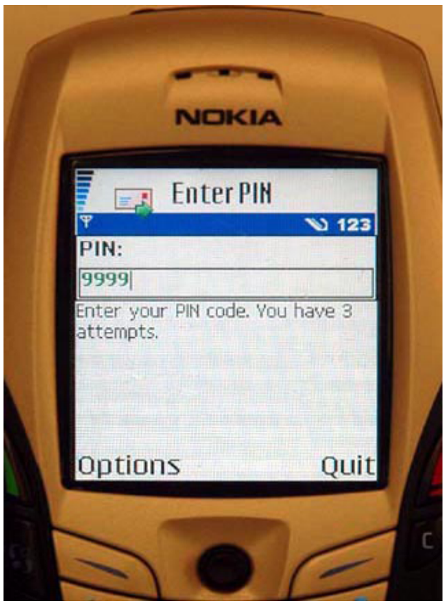


Figure 19: Nokia 6600 - PIN screen 1

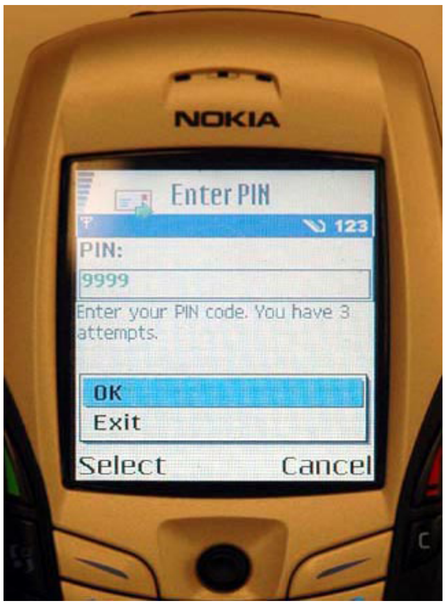
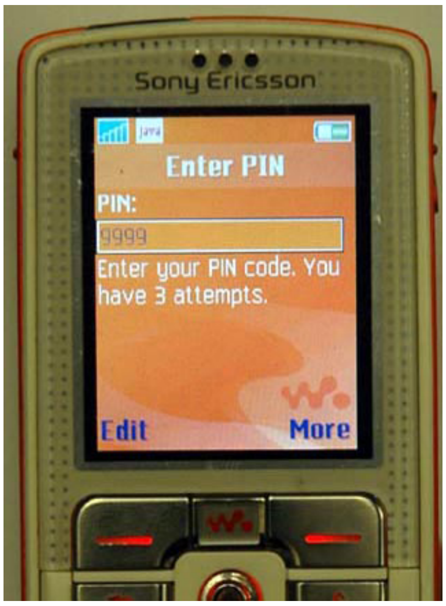
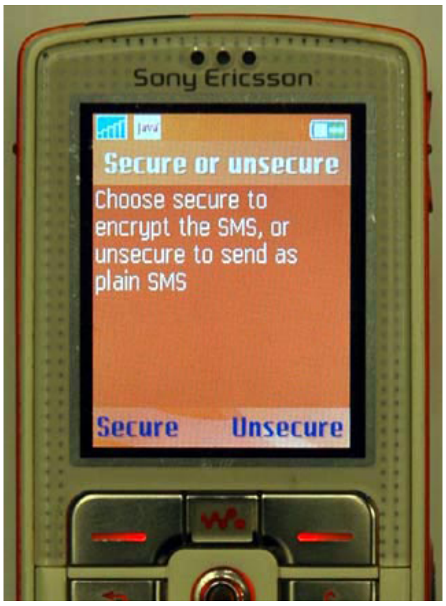


Figure 20: Nokia 6600 - PIN screen 2



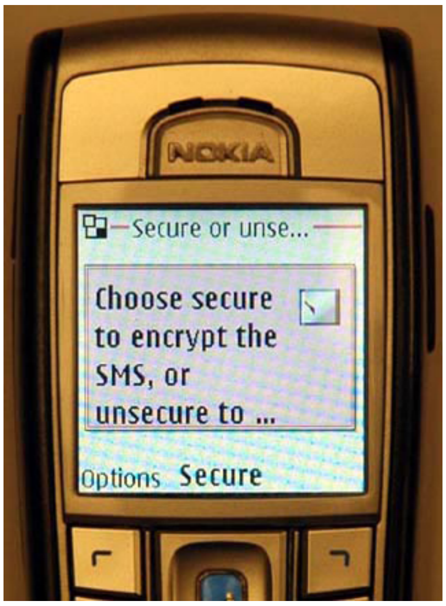
Enter your PIN code. You have 3 attempts.

Figure 21: SonyEricsson W800 - PIN screen 3



Choose secure to encrypt the SMS, or unsecure to send as plain SMS

Figure 22: SonyEricsson W800 - Send Secure/Unsecure



Choose secure to encrypt the SMS, or unsecure to ...

Figure 23: Nokia 6230 - Send Secure/Unsecure 1/2

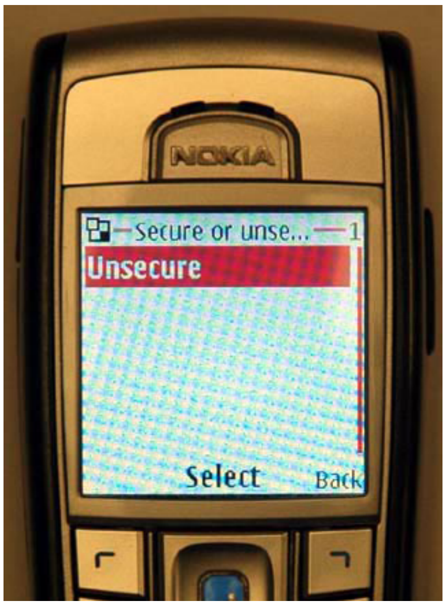


Figure 24: Nokia 6230 - Send Secure/Unsecure 2/2

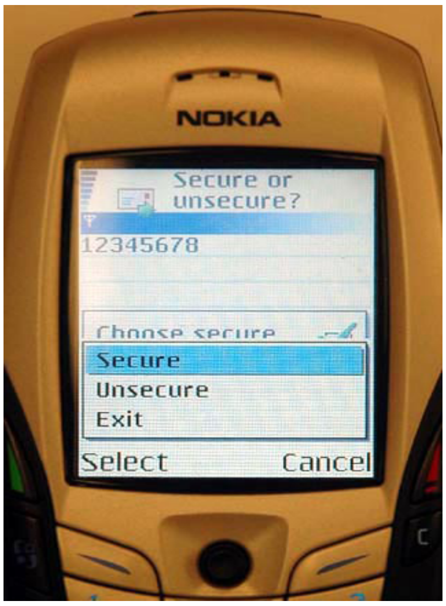


Figure 25: Nokia 6600 - Send Secure/Unsecure