

System Integrity for Smartphones

Master thesis performed in Information Coding
by

Fredrik Hansson

LiTH-ISY-EX--11/4494--SE

Linköping 2011-09-04

System Integrity for Smartphones

A security evaluation of iOS and BlackBerry OS

Master thesis in Information Coding
at Linköping Institute of Technology
by

Fredrik Hansson


LiTH-ISY-EX--11/4494--SE

Supervisor: Thomas Abrahamsson (Sectra Communications AB)

Supervisor: Viiveke Fåk (Department of Electrical Engineering)

Examiner: Viiveke Fåk

Linköping 2011-09-04

Presentation Date 2011-08-29 Publishing Date (Electronic version) 2011-09-30	Department and Division Department of Electrical Engineering	 Linköpings universitet
---	--	--

Language <input checked="" type="checkbox"/> English <input type="checkbox"/> Other (specify below)	Type of Publication <input type="checkbox"/> Licentiate thesis <input checked="" type="checkbox"/> Degree thesis <input type="checkbox"/> Thesis C-level <input type="checkbox"/> Thesis D-level <input type="checkbox"/> Report <input type="checkbox"/> Other (specify below)	ISBN (Licentiate thesis) ISRN: LiTH-ISY-EX--11/4494--SE Title of series (Licentiate thesis) Series number/ISSN (Licentiate thesis)
Number of Pages 87		

URL, Electronic Version http://www.ep.liu.se

Publication Title System Integrity for Smartphones – A security evaluation of iOS and BlackBerry OS
Author(s) Fredrik Hansson

Abstract <p>Smartphones are one of the most popular technology gadgets on the market today. The number of devices in the world is growing incredibly fast and they have today taken an important place in many person's everyday life. They are small, powerful, always connected to the Internet and they are usually containing a lot of personal information such as contact lists, pictures and stored passwords. They are sometimes even used as login tokens for Internet bank services and web sites. Smartphones are, undoubtedly, incredible devices! But are smartphones secure and is stored information safe? Can and should these devices be trusted to keep sensitive information and personal secrets? Every single day newspapers and researcher warns about new smartphone malwares and other security breaches regarding smartphones. So, are smartphones safe to use or are they a spy's best friend in order to surveil a person? Can a user do anything to make the device more secure and safe enough to use it in a secure manner? All these questions are exactly what this paper is about!</p> <p>This paper is addressing two popular smartphone platforms, iOS and BlackBerry OS, in order to evaluate how secure these systems are, what risks that occur when using them and how to harden the platform security to make these platforms as secure and safe to use as possible. Another aim of this paper is to discuss and give suggestions on how a separate and dedicated hardware token can be used to improve the platform security even further. In order to evaluate the security level of these platforms, a risk and threat analysis has been made as well as some practical testing to actually test what can be done. The test part consists mostly of a proof-of-concept spyware application implemented for iOS and an IMSI-catcher used to eavesdrop on calls by using a rogue GSM base transceiver station.</p> <p>The implemented spyware was able to access and transfer sensitive data from the device to a server without notifying the user about it. The rogue base station attack was even scarier since with only a few days work and equipment for less than \$1500 can smartphones be tricked to connect to a rogue base station and all outgoing calls can be intercepted and recorded. The risk analysis resulted in not less than 19 identified risks with mixed severity of the impact. Some configurations and usage recommendation is given in order to prevent or mitigate these risks to make the usage of these platforms safer. The aim of suggesting how a hardware token can be used to strengthening these platforms have been a bit of failure since no really working suggestion has been possible to give. It is a result of that these systems are tightly closed for modification by third parties, and such modifications are needed in order to implement a working hardware token. However, a few partial suggestions for how such a token can work are given.</p> <p>The result of this work indicates that neither iOS nor BlackBerry OS is entirely secure and both need to be configured and used in a correct way to be safe for the user. The benefits of a hardware token should be huge for these systems but the implementations that are possible to do is not enough and it might not be of interest to implement a hardware token for these systems at the moment. Some of the identified risks require the attacker to have physical access to the device and this can only be prevented if the user is careful and acts wisely. So, if you want to use high technology gadgets such as smartphones, be sure to be a smart user!</p>
--

Keywords Smartphone, Integrity, iOS, BlackBerry, Risk analysis, iPhone
--

Abstract

Smartphones are one of the most popular technology gadgets on the market today. The number of devices in the world is growing incredibly fast and they have today taken an important place in many person's everyday life. They are small, powerful, always connected to the Internet and they are usually containing a lot of personal information such as contact lists, pictures and stored passwords. They are sometimes even used as login tokens for Internet bank services and web sites. Smartphones are, undoubtedly, incredible devices! But are smartphones secure and is stored information safe? Can and should these devices be trusted to keep sensitive information and personal secrets? Every single day newspapers and researcher warns about new smartphone malwares and other security breaches regarding smartphones. So, are smartphones safe to use or are they a spy's best friend in order to surveil a person? Can a user do anything to make the device more secure and safe enough to use it in a secure manner? All these questions are exactly what this paper is about!

This paper is addressing two popular smartphone platforms, iOS and BlackBerry OS, in order to evaluate how secure these systems are, what risks that occur when using them and how to harden the platform security to make these platforms as secure and safe to use as possible. Another aim of this paper is to discuss and give suggestions on how a separate and dedicated hardware token can be used to improve the platform security even further. In order to evaluate the security level of these platforms, a risk and threat analysis has been made as well as some practical testing to actually test what can be done. The test part consists mostly of a proof-of-concept spyware application implemented for iOS and an IMSI-catcher used to eavesdrop on calls by using a rogue GSM base transceiver station.

The implemented spyware was able to access and transfer sensitive data from the device to a server without notifying the user about it. The rogue base station attack was even scarier since with only a few days work and equipment for less than \$1500 can smartphones be tricked to connect to a rogue base station and all outgoing calls can be intercepted and recorded. The risk analysis resulted in not less than 19 identified risks with mixed severity of the impact. Some configurations and usage recommendation is given in order to prevent or mitigate these risks to make the usage of these platforms safer. The aim of suggesting how a hardware token can be used to strengthening these platforms have been a bit of failure since no really working suggestion has been possible to give. It is a result of that these systems are tightly closed for modification by third parties, and such modifications are needed in order to implement a working hardware token. However, a few partial suggestions for how such a token can work are given.

The result of this work indicates that neither iOS nor BlackBerry OS is entirely secure and both need to be configured and used in a correct way to be safe for the user. The benefits of a hardware token should be huge for these systems but the implementations that are possible to do is not enough and it might not be of interest to implement a hardware token for these systems at the moment. Some of the identified risks require the attacker to have physical access to the device and this can only be prevented if the user is careful and acts wisely. So, if you want to use high technology gadgets such as smartphones, be sure to be a smart user!

Acknowledgements

This paper is the result of my master thesis in Computer Science carried out at Sectra Communications AB in Linköping. I would like to take the opportunity to show my gratitude to some persons that have contributed to this work, which have made this work possible as well as improving the quality of this work. All their contributions have eased the work for me and without them would the result of this thesis been much less satisfying.

The first person I want to thank is Associate Professor Viiveke Fåk at Linköping University, who were both the examiner and my academic supervisor for this thesis. Viiveke has contributed with factual corrections, improvement suggestions as well as linguistic enhancements, which all have been of great value for me.

The second person I want to thank is Thomas Abrahamsson, who was my corporate supervisor at Sectra Communications AB. He has contributed with ideas and important guidance throughout the work. I also want to send a special thank to Lars Helgeson at Sectra Communications, who took the time to be my substitute supervisor in the absence of Thomas. Lars contributed with time and patient when listening on my problems, which in the time of need was indeed important for me.

There are a bunch of more persons at Sectra Communications that has contributed in different ways that helped me during this work. They are too many to specify by name here, but I want to thank all of them for their help and I hope none of them feel left out.

I also want to thank Jacob Tallberg, who was the opponent at the presentation of this master thesis. He contributed with interesting feedback on my work and with some improvement suggestions for this paper.

Finally, I want to show my gratitude to K.T, who have contributed in a more personal way. She have been supporting, understanding and she have always shown her believe in me. She has made it possible for me to relax and stop focusing on my work at evenings and weekends, which indeed have been important. Without her by my side during this period would this never have been possible.

Thank you all, for your help and your support!

Fredrik Hansson

Linköping, September 2011

Table of Contents

1	Introduction.....	1
1.1	Motivation.....	1
1.2	Aim and purpose.....	1
1.3	Problem formulation.....	2
1.4	Limitations.....	2
1.5	Method and work process.....	2
1.6	Paper structure.....	3
2	Fundamentals.....	5
2.1	What is smartphone integrity?.....	5
2.1.1	Information integrity.....	5
2.1.2	Personal integrity.....	5
2.1.3	Company integrity.....	5
2.1.4	Smartphone integrity.....	6
2.2	Risk categories.....	6
2.2.1	Data leakage.....	6
2.2.2	Surveillance.....	6
2.2.3	Unavailability	6
2.2.4	Financial.....	7
2.3	Attack categories.....	7
2.3.1	Physical access attacks.....	7
2.3.2	Remote access attacks.....	7
2.3.3	Malware attacks.....	8
2.3.4	Social engineering attacks.....	8
3	iOS platform.....	9
3.1	Architecture.....	9
3.1.1	Application layer.....	10
3.1.2	Cocoa Touch layer.....	10
3.1.3	Media layer.....	10
3.1.4	Core Service layer	10
3.1.5	Core OS layer.....	10
3.1.6	Hardware layer.....	11
3.2	Security features.....	11
3.2.1	User authentication.....	12
3.2.2	Device encryption.....	12
3.2.3	Data protection.....	12
3.2.4	Keychain.....	13
3.2.5	Sandbox.....	13
3.2.6	Address space layout randomization.....	13
3.2.7	Secure network communication.....	13
3.2.8	Backup and remote wipe.....	14
3.3	Enterprise policies.....	14
3.4	Application distribution.....	15
3.5	Baseband.....	15
3.6	Bootloader.....	16
4	BlackBerry OS platform.....	17
4.1	Architecture.....	17
4.1.1	BlackBerry Internet Service.....	17
4.1.2	BlackBerry Enterprise Server.....	18
4.1.3	Device architecture.....	19
4.2	Security features.....	20
4.2.1	User authentication.....	20
4.2.2	Data encryption.....	20
4.2.3	Communication security.....	20
4.2.4	Sandbox.....	21

4.2.5 Memory clean and device wipe.....	21
4.3 Key management.....	21
4.4 IT-policies.....	23
4.5 Application distribution.....	24
4.6 Baseband.....	24
4.7 Bootloader.....	24
5 Proof-of-concept spyware.....	27
5.1 Accessible data.....	27
5.2 Behavior obfuscating.....	28
5.3 Conclusions.....	29
6 Rogue GSM base station.....	31
6.1 Attack description.....	31
6.1.1 GSM limitations.....	31
6.1.2 Attack setup.....	32
6.1.3 Legal restrictions.....	32
6.2 Attack equipment.....	33
6.2.1 BTS hardware.....	33
6.2.2 BTS software.....	33
6.2.3 Other equipment.....	34
6.3 Modifications.....	35
6.4 Test result.....	35
6.4.1 Connection.....	35
6.4.2 Calls.....	35
6.4.3 SMS.....	36
6.4.4 Over-the-air-programming.....	36
6.5 Conclusions.....	36
7 Risk analysis.....	39
7.1 Analysis fundamentals.....	39
7.1.1 Assets.....	39
7.1.2 Agents.....	39
7.1.3 Threats.....	39
7.1.4 Vulnerabilities.....	39
7.1.5 Complexity.....	40
7.1.6 Impact.....	40
7.1.7 Risks.....	40
7.2 Method and work process.....	40
7.3 Asset Identification.....	42
7.3.1 Data assets.....	42
7.3.2 Hardware assets.....	42
7.3.3 Firmware assets.....	43
7.3.4 Communication assets.....	43
7.3.5 Financial assets.....	43
7.3.6 Service assets.....	43
7.4 Threat identification.....	43
7.4.1 Data leakage.....	43
7.4.2 Financial loss.....	44
7.4.3 User surveillance.....	44
7.4.4 Device and/or service unavailable.....	44
7.4.5 Credentials leakage.....	44
7.5 Attack identification.....	44
7.5.1 Jailbreaking/Rooting.....	45
7.5.2 Malware attacks.....	45
7.5.3 Phishing attacks.....	46
7.5.4 Attacks on encryption.....	46
7.5.5 Attacks on GSM/3G.....	47
7.5.6 Hardware tampering.....	47
7.5.7 Spoofed Wi-Fi access points.....	48

7.5.8 Network attacks.....	48
7.5.9 Theft.....	49
7.6 Risk identification.....	49
7.7 Analysis result.....	52
8 Solutions and mitigations.....	53
8.1 Jailbreaking/Rooting.....	53
8.2 Malware attacks.....	53
8.3 Phishing attacks.....	54
8.4 Attacks on encryption.....	54
8.5 Attacks on 3G/GSM.....	54
8.6 Hardware tampering.....	55
8.7 Spoofed Wi-Fi access points.....	55
8.8 Network attacks.....	56
8.9 Theft.....	56
9 Hardware token.....	57
9.1 Dream scenario.....	57
9.1.1 Dream utilization.....	57
9.1.2 Platform limitations.....	58
9.2 Token suggestion.....	58
9.2.1 Utilization.....	58
9.2.2 Authentication.....	59
9.2.3 Reusability.....	60
9.3 Conclusions.....	60
10 Results and discussion.....	63
10.1 Results.....	63
10.2 Discussion and conclusions.....	63
10.3 Future work.....	65
11 Definitions and abbreviations.....	67
12 References.....	73
13 Appendixes.....	77
13.1 Appendix A: iOS layer and frameworks overview.....	77
13.2 Appendix B: BlackBerry encryption key hierarchy.....	79
13.3 Appendix C: OpenBTS modification.....	81
13.4 Appendix D: Risk tables.....	83

Index of Figures

Figure 1: iOS layers.....	9
Figure 2: iOS kernel overview.....	11
Figure 3: iOS bootloader usage.....	16
Figure 4: BlackBerry Internet Service Overview.....	17
Figure 5: BlackBerry Enterprise Solution Overview.....	18
Figure 6: Blackberry OS layers.....	19
Figure 7: Code for accessing keyboard cache.....	28
Figure 8: Code for accessing keyboard cache with string obfuscating.....	28
Figure 9: Spy categories.....	29
Figure 10: System information.....	29
Figure 11: Keyboard cache.....	29
Figure 12: Overview of the attack scenario.....	32
Figure 13: The attack equipment: BTS and laptop.....	36

Index of Tables

Table 1: Example of configuration profiles categories.....	15
Table 2: Encryption keys on a BlackBerry device.....	22
Table 3: List of used hardware for rogue BTS.....	33
Table 4: List of used software for rogue BTS.....	34
Table 5: Complexity and impact levels in this analysis.....	41
Table 6: Risk levels in this analysis.....	42

1 Introduction

We are living in a very technological society and the majority of persons have access to mobile phones and computers every single day. One of the most popular and fastest growing types of devices is smartphones. A smartphone is a mobile phone combined with the power of a small computer and a smartphone stays connected to Internet wherever we go. We can check our emails, browse the Internet, call our friends and much more on a device that is small and easy to bring. Smartphones seem to make our lives much easier regardless if we use them for work purposes or just for fun in our private lives. But how secure are smartphones? Can they be hacked and can I lose my private information to some evil attacker? Can someone eavesdrop on me and my company through my smartphone? These are questions that most smartphone users do not think about before it might be too late. This thesis will address the platforms iOS and BlackBerry OS in order to evaluate their strengths and weaknesses and give suggestions for solutions and mitigations in order to improve the security in these platforms.

1.1 Motivation

Smartphones are very popular and more and more people are using them every day. The number of smartphone users is increasing in great speed and it is predicted that this trend will keep going for at least a couple of years. Hence, we will see a lot of smartphones in the world in the future. For every new smartphone model that enters the market the power is increased and each model is more sophisticated than the previous one. Today's smartphones are already as powerful as a small computer and can do a lot of the things a computer can do, i.e. browse Internet, send emails and play games and much more. Besides the power of the device, it contains a lot of important and in some cases sensitive information, such as images, contact information, addresses and calendars. The usability together with the size and mobility of these devices makes users bring their devices wherever they go and users almost always have their devices turned on. So users are always carrying around a powerful computer that is connected to the Internet and that is stuffed with sensitive and private information.

Today, the understanding for smartphone security is quite low compared to the understanding for computer security. Almost every computer user knows that there are nasty malwares and evil hackers in the computer world and that they must protect their computers from such threats. It applies to smartphones as well but users are still not alerted enough to see the need for security tools for smartphones. Actually, it is said that security awareness for smartphones is at the same level that it was for computer security in the early 1990's. The lack of security awareness opens up for a lot of different attacks on smartphones and their users. For instance, an attacker can eavesdrop on a user through the device, all information on a device can be read, copied and/or erased by an attacker and an attacker can make expensive calls from the smartphone which lead to a very expensive bill for the user and so on.

There are a lot of different operating systems for smartphones and they have different architectures and features. The differences between systems make each system work in their own specific way, which leads to different vulnerabilities in each system even though some vulnerabilities exist in more than one operating system. The most popular operating systems for smartphones at the market today are: Apple's iOS, Google's Android OS, RIM's BlackBerry OS, Microsoft's Windows Mobile 7 and Symbian's Symbian OS.

1.2 Aim and purpose

The aim of this thesis is to investigate and evaluate how secure the smartphone platforms iOS and BlackBerry OS are. These platforms are chosen for review since they are considered to be among the most secure smartphone platforms on the market. This thesis is also about finding solutions for any identified weakness in these platforms so that smartphone integrity can be ensured. Hopefully will this paper be an eye-opening reading for the common smartphone users that perhaps not yet have thought about the existing security risks with their device and how a user's behavior contributes to vulnerabilities and risks of the integrity of smartphones.

1 Introduction

Another aim for this thesis work is to come up with suggestions for a hardware token that can improve the security for these platforms and can be used to authenticate the user. A major interest is that this token should be working for at least both iOS and BlackBerry OS but it is also desired that it could be used with other systems as well.

1.3 Problem formulation

The following list shows the problems and questions that this thesis addresses:

- How strong are the existing platform security frameworks in iOS and BlackBerry OS?
- What are the weaknesses of the iOS and BlackBerry OS platforms?
- What solutions can be made to ensure the integrity for these platforms?
- How can a separate and dedicated hardware token be utilized to improve the security in these platforms?
- How can a user be authenticated using the utilized hardware token?
- Can the found security solution concept be used on other operating systems and platforms?

1.4 Limitations

This thesis work has a few limitations in order to keep the extent of this thesis work to a reasonable size. First of all, this work is limited to only include Apple's iOS and RIM's BlackBerry OS and no other platforms will be studied in this work.

Another limitation is that only the latest versions of each platform will be evaluated. Hence, no previous versions and no old vulnerabilities (which do not exist in the current version) will be discussed in this thesis work. The current version of iOS is 4.3.3 and the current version of BlackBerry OS is 6.0.

The last limitation of this thesis work is that only unmodified devices are included in this analysis. Therefore, iOS devices that are jailbroken and BlackBerry devices that are rooted are not discussed in this thesis work since the intention of this work is to evaluate the security level of the platforms in original state and not after they are modified by the user to be less secure. See for section 7.5.1 more information about jailbreaking/rooting.

1.5 Method and work process

Since this has been a mostly theoretical work, one major part of it was to find, read and summarize information from different sources. The collected information has been used as a foundation for the security evaluation that has been performed. First each platform was studied to evaluate their strengths and weaknesses. When enough background information had been studied, the next part was to perform the risk and threat analysis for each of these platforms. The chosen method for the risk analysis was a multistep processes including the steps: asset identification, threat identification, attack identification, risk identification and vulnerability identification. The impact of each threat and the complexity of each attack are evaluated in order to measure the severity of each risk. When threats, risks and weaknesses were found for each platform, the next part of this work was to find solutions and mitigating measures for each weakness. The last thing was to study and come up with some ideas on a hardware token that could improve the security and integrity of these smartphone platforms.

Even though this has been a mostly theoretical study, it was of interest to perform practical test of the platform security by performing some penetration tests and similar tests, e.g. port scanning and so on. A practical part including a test attack on the GSM-network has been included as well. A small proof-of-concept spy application for iOS has been developed to conceptually evaluate what resources that can be accessed from a malicious application running on iOS.

1.6 Paper structure

This paper is structured in the following way; the first part discusses the background information needed for understanding the rest of the paper. An important thing in this part is the definition and explanation of the term “*smartphone integrity*” which this entire thesis is about. The following parts describe the two addressed platforms in great detail, i.e. iOS and BlackBerry OS. The architecture and security features of these systems are, among other things, discussed.

After the description of these systems, two chapters which discuss the practical test part follows. These chapters address the implemented proof-of-concept malware and the GSM-network attack. After that comes the major part describing the risk analysis and its result. The risk analysis part is the biggest and it might feel heavy, but it is also the most important chapter together with the following one, which contains the description of recommended mitigations that can make these systems more secure. After the mitigations is the utilized hardware token described and discussed. The last major part is the result and discussion part of this paper. At the end of this paper there is a list of important definitions and abbreviations that can be consulted if the reader does not understand some concepts that are mentioned in the text.

An important thing to notice is that this paper is written and structured in such a way that it is strongly recommended that a reader reads it from the beginning to the end and does not jump back and forth. All parts are important for the reader to be able to follow the red line throughout this paper.

2 Fundamentals

This chapter explains and defines some of the most important concepts that must be understood for a reader to be able to follow this paper. It also describes different risk categories and attack categories that are discussed in this thesis work. Some small and often used concepts and abbreviations can be looked up in the definition list in chapter 11 if the reader does not have knowledge about them.

2.1 What is smartphone integrity?

Smartphone integrity is still a kind of new concept and it is perhaps not trivial to understand exactly what it means. Actually, it is not easy to put a definition on the concept either. Smartphone integrity is a wide concept and it seems to have different definitions depending on who is explaining it. So, in this section the definition of smartphone integrity as used in this thesis work is explained. Definitions for some other types of integrity are first described as background theory for smartphone integrity.

2.1.1 Information integrity

Integrity in information security is usually defined as the concept that information cannot be modified or tampered with without being noticed. Hence, it is not about keeping the information secret or inaccessible for an attacker. It is the concept that an attacker should not be able to modify or tamper with the information without it being noticed by the information owner. It is also about knowing that information actually comes from the origin that it claims to be from. A user should therefore be able to trust that information is valid and unchanged every time he wants to use it [1]. Below follows an example scenario of breaking information integrity:

A person writes down a friend's account number on a piece of paper because he should transfer money to the friend. Next time he reads the note a third person has changed the account number to his account number. It leads to that the transfer ends up at the third person's account and not at the friend's account.

2.1.2 Personal integrity

Integrity is often mentioned in everyday life in the sense of personal integrity. Personal integrity means that personal and private information should not be leaked to someone else. In one sense it can be defined as that no information about a person and the person's private life should be known by anyone that person does not want them to know. Of course are things the person says included in this, i.e. a person should be able to have a conversation without having someone breaking his integrity by eavesdropping. Another important aspect of personal integrity is that no one should be able to act like the person and do things in his name. An example scenario of breaking the personal integrity is:

A person talks with his colleague in the lunchroom about his relation problems. At the next table sits another colleague and eavesdrop on them. The eavesdropping colleague later tells the entire company about what he heard and in this way breaking that person's integrity.

2.1.3 Company integrity

The last type of integrity that is important in this case is company integrity. Not only individuals should have their integrity, it should hold for companies and groups of persons as well. Most companies have information that should not be accessible for persons outside the company. So, the integrity of a company should be guaranteed so that the company can trust that information does not end up somewhere they do not want it to be. It is also important that no other company or person can act as and pretend to be the company and do things in the name of the company. An example scenario of breaking company integrity could be:

A person pretends to be the spokesman of a car company and tells the media that their latest cars should not

2 Fundamentals

be bought because they are dangerous to drive. This hurts the company's reputation by breaking the company integrity.

2.1.4 Smartphone integrity

What does smartphones have to do with the above stated concepts? The first thing smartphone users should realize is that they probably have lots of personal and private information stored on their smartphones. Information and data that users probably do not want someone else to be able to access.

Many smartphones are used inside companies, as work phones or employee's personal phones, and eavesdropping on such devices is a very good way for an attacker to get hold of company secrets. A smartphone can also be used by hackers as a way to access internal networks inside companies.

So, what is smartphone integrity then? It is a combination of the above described types of integrity. Therefore, smartphone integrity is the concept of ensuring that personal and private information, as well as any company confidential information, cannot be accessed from the device by any unauthorized persons. It should also ensure the privacy of communications by preventing eavesdropping. Communication in this sense includes phone calls, e-mails, messages and conversations close to the device and so on.

It is also about information integrity and device integrity since it must be detectable if information on the device or the device itself has been modified and/or tampered with. This is important so a tampered device can be taken out of use or at least be fixed. To summarize this can it be said that:

Smartphone integrity is the concept of keeping the information on a smartphone secure and the use of a smartphone safe to protect the integrity of the user.

2.2 Risk categories

This section explains the different types of risks with smartphone usage that are addressed in this paper. There are four main categories of risks that are addressed and all identified risks are categorized in one of these four categories. Below follows the explanation for each of the four main categories:

2.2.1 Data leakage

Data leakage is one main category for risks with smartphones. As stated before, the average smartphones contains a massive amount of data. Everything from emails, phone numbers, calendars to company confidential files and passwords are often stored on smartphones. These data are in many cases extremely sensitive and should not end up in the hands of an attacker. Data leakage is about just that, information leaks in some way from the device into the hands of an attacker. Data can leak in many different ways e.g. through stolen devices, malware copying files and so on [2].

2.2.2 Surveillance

Surveillance is another main category for risks and it is about spying on users. Smartphones often have GPS-modules available today and therefore it is possible to get the exact latitude and longitude coordinates of a device. So, it is possible to keep track of the user's exact position at all time. Surveillance is also about collecting information about the behavior of users [2]. Surveillance information can be which time the user get up in the morning, how the user uses his smartphone, e.g. does the user browse the Internet with the phone? If so, which pages does he often visits? Surveillance also includes interception of calls and messages so that an attacker can eavesdrop on the user's conversations and doings.

2.2.3 Unavailability

Unavailability is the category for risks that a user will not be able to use the smartphone or a service when he needs it [2]. It might be hard to see this as a severe problem for a user, but for the case of where the device is

used for business matters can this indeed be a severe problem, e.g. a doctor on call that cannot be reached. Unavailability of a device can be done by stealing the device, have a malware that drains the battery fast, destroying the device and so on.

2.2.4 Financial

Financial is the last main category for risks that will be discussed in this paper. Smartphones are in a very tight way connected to users' finances in the sense of phone bills. Mobile phone accounts are used more and more as a payment method for services by charging the user through the phone bill. It is therefore a risk that an attacker tries to earn money for instance by calling expensive numbers which he later will get money from or an attacker just makes expensive calls to hurt the user's finances [2]. Smartphones are today used more often as an authentication device and it can therefore be a risk that other systems can be accessed with help of the smartphone. If such systems can be used for buying things it will be a financial risk for the user as well. Examples of attacks that lead to financial risk are a thief stealing the device, malware making expensive calls and/or sending SMS and so on.

2.3 Attack categories

There exists a lot of possible ways for an attacker to attack a smartphone system. In this paper are all attacks categorized in one of four main categories and these main categories are described and explained in this section. It might be difficult to place some attacks just in one of this categories since an attack can consist of more than one step and each step can be using different types, e.g. an attacker uses a remote access to a device over internet, and to open a backdoor the attacker uses a malware that was installed on the device. It can be sufficient to prevent one step to make a multi-step attack to fail but in some more complex attacks more than one step must be prevented to stop the attack. For instance, in the above given example it should be sufficient to prevent the malware from opening the backdoor to stop the entire attack.

2.3.1 Physical access attacks

Physical access attacks is one main category for attacks and it includes all attacks that are performed while having physical access to the device, i.e. theft of device, manually installing malicious applications, changing smartphone settings, modify the hardware and so on. The exposure time of the device is very important for the outcome of this type of attacks. Letting a device out of sight for just a few seconds is enough for a thief to steal it but it is not enough with a few seconds for an attacker to read sensitive information on the device or modify the hardware [2]. A possible attack scenario could be:

A person checks his emails on his smartphone while eating dinner at a restaurant. He leaves the smartphone at the table and goes to the bathroom. An attacker sitting at the next table takes advantage of the minutes alone with the device and quickly replaces the battery on the device for another battery that looks the same but contains a microphone. So, now the attacker can eavesdrop on the person.

2.3.2 Remote access attacks

Remote access is another main category for attacks and all attacks that are performed without physical access to the device are included in this category, e.g. using known vulnerabilities to hack into the device over internet, remotely turn on the microphone to eavesdrop on the user and so on. This type of attacks are often depending on some other attack from another category to be successful, i.e. using a malware to open a backdoor into the device or trick the user to open the backdoor for the attacker [2]. A possible attack scenario could be:

An attacker uses a known vulnerability in some application installed on a victim's smartphone to access the smartphone remotely over the internet and gain access to the device. The attacker then remotely turns on the microphone on the smartphone and can in this way eavesdrop on the victim's doings.

2.3.3 Malware attacks

Malware attacks are another category of attacks and it includes all attacks that are performed by some malicious code executed on the smartphone. There exist a lot of different types of malwares that can perform a lot of malicious actions, e.g. common malware types are spywares, worms, viruses and trojans [2]. Typical actions a smartphone malware can do are opening backdoors, send sensitive data from the device to the attacker, delete and/or change data on the device and so on. There exist multiple ways to get a malware into the system, e.g. by email, by SMS/MMS, from Internet files, inside applications, by Bluetooth transfers and so on. A possible attack scenario could be:

An attacker writes a spyware and disguises it as a game application and distributes it through the Internet. The victim downloads the game and plays it. While the victim plays the game the application copies the entire address book on the device and uploads it to the attacker's server.

2.3.4 Social engineering attacks

Social engineering attacks are the last category of attacks this paper focuses on. Social engineering attacks are attacks against the human user and not directly an attack on smartphones. Social engineering is a kind of old fashioned swindle to get the user to do something for the attacker [1]. In the case of smartphone integrity social engineering attacks probably mostly are for tricking a user to install a malware, changing settings so the attacker later can remotely access the device, tricking the user to enter his passwords in a spoofed web site or something similar. So, this kind of attack is often just a first part of an attack and not a final attack in this sense. A possible attack scenario could be:

An attacker sends an email to a victim pretending to be an application developing company. The email states that the victim has been chosen to be one of few persons that can get the company's latest cool application for free. The email also includes a link for the victim to follow for downloading the application. The victim happily clicks on the link to download the application and at the same time he is downloading the attacker's spyware into his smartphone.

3 iOS platform

iOS is an operating system developed by Apple Inc. to be used in their mobile devices such as iPhone, iPad and iPod Touch, (note that only iPhone is a smartphone). The first version of iOS was released in June 2007 together with the first model of iPhone. iOS has since then regularly been updated and at the moment is the latest version of iOS 4.3.3. At their homepage Apple proudly state that iOS 4 is:

“The world's most advanced mobile operating system” [3].

This operating system is indeed advanced and it has at the same time a very user-friendly layout and design, which makes it easy to use for non-technical users. The operating system is closed in the sense of no available source code and it is not free to develop applications for this system. To be able to develop and test applications on a device running iOS the developer must register a developer license with Apple and the license must be renewed every year.

The source code for the operating system is not available and Apple has a policy of not commenting on how their systems work or behave and it is therefore not trivial to actually know the structure of this system. This can be considered as both good for the security and bad for the security of the system. It can be good since it is harder for an attacker to find any weaknesses of the system but it can also be bad since there are less people analyzing the system and therefore it will probably take longer time before weaknesses are discovered and can be fixed [4]. An import thing to remember is that relying on keeping the system structure secret is in no way a good security solution, but if the system is well protected and reviewed it is not bad for the security to keep the specification secret. To rely on secrecy to achieve security in a system is called security by obscurity and security experts have considered it as a disapproved method for a long time. The rest of this chapter discusses the architecture, structure and security features of the iOS platform.

3.1 Architecture

The architecture of the operating system iOS can be divided into six different layers and each layer has its own structure and purpose. The layers are shown in Figure 1 below. In the following sections each one of the layers is briefly described and its purpose for the system is pointed out. It starts in the higher level of the operating system and goes downwards to the hardware at the bottom. Since the Core OS layer is the layer that is most important to understand for the security aspect in iOS, this layer will be described in greater detail than the rest of the not so important layers. Most layers provide a bunch of frameworks for accessing different information and services of the operating system. An overview of the layers and all their frameworks is given in Appendix A: iOS layer and frameworks overview.

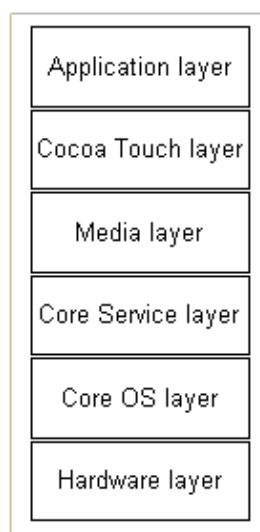


Figure 1: iOS layers

3.1.1 Application layer

The application layer is the top layer in the iOS architecture and it represents the applications running in this system. This layer contains the application's internal environment and its part of the memory. Even though this layer is the top layer in the architecture model it must be understood that it can access many of the underlying layers and not only the closest one. What can be done in this layer is heavily depending on the application. Some applications are games, others are email clients and so on and of course different kind of applications are working in different ways and are using different underlying frameworks.

3.1.2 Cocoa Touch layer

Cocoa Touch is the second layer from the top and it contains a bunch of frameworks that often are used in applications. The use of these frameworks is mostly for creating and handling user interfaces but there are a few other uses for them as well. The largest and most used framework in this layer is the UIKit framework which takes care of application features such as user interface management, application event handling, data handling, hardware library interaction and much more. All running applications are connected to this layer since applications are at least using the UIKit framework [5]. This layer is indeed very important for an application developer to understand but it will not be discussed any further in this paper.

3.1.3 Media layer

The media layer contains all frameworks that are required to handle graphics, animations, video and audio capabilities in applications. All applications that play any sound, video, animation or handle anything else than text-based graphics must be connected to this layer and use its services to provide user interface graphics. So, this layer is mostly for providing a rich user experience in sense of graphics, video and audio and therefore it is not discussed any closer in this paper. [5] contains a much more detailed description of this layer and its frameworks.

3.1.4 Core Service layer

The core service layer is one of the low-level layers in the iOS architecture and it provides frameworks and API:s for accessing and modifying so called core services, i.e. address book, GPS-location of the device, system configuration and so on. According to documentation [5] using this framework is the only way to access core services, and therefore should the access to these services be limited for security reasons. For instance, it should not be possible to copy the entire address book of a device and upload it to a server from within a third party application. This layer includes the three security services that iOS has. Those are the Keychain Service, the Certificate, Key and Trust Service and the Randomization Service. These services are accessed from applications through the security framework described in the Core OS Layer section.

3.1.5 Core OS layer

The core OS layer is the actual core of the operating system and it provides a few low-level frameworks for handling core features. Since this is the core of the operating system it contains the kernel and it handles all things that must be managed by the kernel. Figure 2 below shows an overview of the kernel architecture in iOS and the different internal systems the kernel handles.

The kernel is based on Apple's Mach-kernel used in all Mac desktops and laptops. However, this Mach-kernel version is slightly modified to be optimized for a mobile environment. The kernel contains systems such as memory system, file system, thread system, IPC system and I/O system. It also contains the low-level network structure and all the drivers required for the underlying hardware. The memory system handles everything associated with both the real and virtual memory of the device. Typical things handled by the memory system is allocating and freeing memory used by applications. The file system handles all access to the files stored on the device. The file system is also responsible for handling the permissions for file access for the different parts of the system and for the applications.

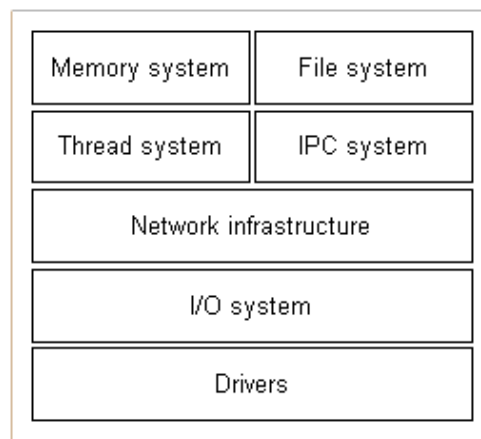


Figure 2: iOS kernel overview

The thread system provides the possibility to execute code in parallel by using threads. It also provides support for synchronization that is needed when executing code in parallel. Since threading is available the kernel must support communication between threads and therefore is the IPC system a part of the kernel. The IPC system provides communication between threads for the operating system. The I/O system handles all input and output on the device.

The network structure of the kernel handles everything that belongs to networking. It handles sockets, packet filtering, network stacks and much more. The last part of the kernel is drivers, which include all the hardware drivers required for both the internal hardware and the external equipments that are connected to the device. Also some required software drivers and filter drivers are included there.

This layer provides a few low level frameworks as well. One important framework is the security framework. iOS provides some built in security features, which will be described later in this chapter, and this framework provides interfaces for this extra security functionality for application data. It provides among other things interfaces for handling certificates, encryption keys and pseudorandom numbers in applications as well as a bunch of encryption algorithms and cryptographic hash algorithms. This framework can make an application much more secure by securing its internal data through encryption and it should therefore be used when developing applications [5].

3.1.6 Hardware layer

The hardware layer is the lowest layer in the iOS architecture and it represents the actual hardware that iOS is running on. The latest iOS is running on an Advanced RISC Machine (ARMv7) architecture which is a 32-bit high performance processor architecture that is highly power efficient. All current iOS devices use the so-called Apple A4 processor, which is developed by Samsung and is a combination between an ARM Cortex-A8 CPU and a PowerVR GPU. The clock frequency of the CPU is 1 GHz [6]. Apple A4 uses two levels of caches, the first level is a 32KB instruction and a 32KB data cache and the second level is a 640KB cache. The A4 processor uses a thirteen stage superscalar pipeline to achieve high performance.

The ARM architecture uses a non-executable stack, which means that data/code pushed on to the stack is unable to be executed. This is used in order to prevent buffer overflow exploits in the sense of that the injected code cannot be executed, i.e. it does not prevent overflows, it just make it harder to exploit this vulnerability [7].

3.2 Security features

The iOS system and the use of it is said to be safer and more secure than many other similar products on the market. Apple seems to have strong confidence in the security of their product and they state at their

3 iOS platform

homepage that iOS 4 is:

“Safe and secure by design” [3].

It is a quite strong statement and it might be hard to ensure that it really is secure. Anyway, iOS includes some extra security features to achieve a higher level of security. Such features include sandboxing, device encryption and much more. The security architecture of iOS and its extra security features are described in this section.

3.2.1 User authentication

User authentication is not turned on by default on iOS devices but it can easily be turned on. iOS supports the traditional way for user authentication by using a passcode/password. It is up to the user to decide which level of security he wants for authentication on his device. A user can choose between the choices: no authentication, passcode or password. The difference between passcode and password is that passcode only is a four-digit code while password can be a string of a lot more than four characters. Passcode is in fact the same as a PIN, which is commonly used for credit cards. In passwords uppercase letters as well as lowercase letters, digits and special characters can be used to create a much stronger password than the passcode is. (Only using a passcode/password actually does not authenticate the user, it only tells the device that the user should be granted access to the device) [8].

The user can also chose if the device should lock itself after a certain time and if so, how long time the device should wait before locking itself. It is also possible for the user to turn on an upper limit for the number of wrong passcode/password entered in a row. If this feature is turned on the device will erase all device data if a user enters the wrong passcode/password ten times in a row.

3.2.2 Device encryption

iOS provides a default enabled device encryption for the entire device while the device is locked. It uses the embedded encryption hardware in the device to encrypt all data stored on the device. The algorithm used is AES and it uses 256-bit keys to ensure secure encryption. The device generates all encryption keys automatically and they are generated only from data available on the device, i.e. the keys do not depend on the passcode/password chosen by the user [9].

AES with a 256 bit key is still considered to be very secure when strong keys are used even against attackers that have access to great computation power. The keys are generated by the system without any user interaction, and the system generates randomized and strong keys, so the keys should be secure enough. Hence, this device encryption should provide a strong protection for data stored on the device.

3.2.3 Data protection

Besides device encryption iOS does provides a limited data protection feature, which uses the hardware encryption to encrypt data and files on the device. While the device is locked it is not possible for any applications to access the encrypted data. The device generates the encryption keys and the keys are stored protected by the passcode/password on the device [10].

The reason that this feature is called limited is that not all data on the device is encrypted. By default are only emails and attachments belonging to the mail application protected by this feature. It is not possible to turn data protection on for other system data either. However, application developers can turn on data protection for their application data if they like to take advantage of this feature. The developer must then specify exactly which data that should be protected [11]. It is important to understand that no system data except emails are protected, i.e. contacts, pictures, calendars and so on are still stored unencrypted (besides the device encryption).

This feature uses the passcode/password of the device to protect the generated encryption keys and it is therefore required to have passcode/password turned on for the device to use data protection. Hence, the data

protection is not more secure than the device passcode/password is, so using a passcode or a weak password is making this data protection much less secure.

3.2.4 Keychain

Keychain is a security feature used in iOS to store passwords, encryption keys and certificates on the device in a secure manner. The keychain is an encrypted container stored in the device memory. All applications get its own part of the keychain to store their own sensitive data in. iOS only grants access for an application to its own part of the keychain, so keychains can not be shared by applications and no application can access another application's keychain items. Neither is there a way to stop an application from having access to its own keychain items in iOS [8].

The keychain in iOS is encrypted using the encryption algorithm Triple-DES with different keys for different parts of the keychain. Some parts of the keychain are encrypted with system generated encryption keys and these are not depending on any user input, but other parts use keys generated from the user's passcode/password. Examples for keychain items that are protected with keys generated from the passcode/password are IMAP-accounts, Google Mail-accounts and saved web site passwords. Examples for items protected with only system-generated keys are Wi-Fi passwords, VPN-passwords and voicemail passwords [9].

3.2.5 Sandbox

Sandboxing is a technique where each running processes is separately encapsulated in a controlled part of the environment with highly restricted permissions and access rights. This is done in order to shield system resources and data from processes and in this way effectively limit the amount of accessible data for processes. A sandbox also encapsulates and protects each process from other processes. This leads to that the process only has tightly controlled read and write access to data and resources in the system. Apple has developed their own sandbox model called Apple Sandbox, which is used in iOS to encapsulate processes [12]. So all applications are running inside their own sandbox, which separates them from other applications and system resources in iOS.

This leads to that applications only can access their own files and preferences directly. If the application needs to have access to system resources, e.g. network structure or some I/O device, the application must call one of the available public API:s to access the resource that way. So, applications should never be able to directly access system resources and can therefore only perform commands that are allowed in the system and are provided by the API:s [8].

3.2.6 Address space layout randomization

Address space layout randomization, ASLR, is one of the newest security features added to iOS. It improves the security of the system and helps preventing jailbreaking. Jailbreaking is the process of getting full root access for the device and in this way unlocks all capabilities in the system. This makes it possible to bypass the implemented user limitations in the system, e.g. application whitelisting. ASLR was first added in the 4.3 version of iOS, which is the latest major version at the moment. ASLR is about randomizing the address space in the memory every time a process is executed so it is not located at the same address every time the process is running. Many attacks require that the attacker knows the address space the process is running in to be successful and therefore the ASLR method can make it much more difficult for an attacker to successfully attack the system [13]. ASLR does not prevent the attack entirely it just makes it much harder to exploit such vulnerabilities that require the address space to be known, e.g. buffer overflows.

3.2.7 Secure network communication

iOS has built in support for secure network communication using both SSL/TLS and VPN. SSL/TLS can be implemented in all applications by using functions in the provided CFNetwork framework. SSL/TLS is also

3 iOS platform

turned on by default in applications like Safari, Mail, and App Store and is automatically used whenever it is supported on the server [8].

There are no available frameworks to programmatically open up VPN-connections within applications, but VPN is supported in the operating system in the sense of a built in VPN-client. In device settings a user can configure a VPN-connection to connect to a private network. The built in VPN-client has support for use of three different security protocols, i.e. L2TP, PPTP and IPSec. All three protocols have support for password authentication as well as a second alternative. L2TP and PPTP provide the possibility to use a RSA SecurID token and IPSec can use certificates for authentication instead of just passwords for a higher level of security.

3.2.8 Backup and remote wipe

To ensure that important data do not get lost in case of a destroyed or lost device iOS supports creation of backups of the entire device. That includes all messages, settings, pictures, contacts, calendar events and so on. This is done by synchronizing the device with a computer using the provided software iTunes. The user can choose to have the backup encrypted with a key generated from a user chosen password.

In the case of loss of the device or theft it is desirable to be able to remotely erase all data on the device, so that a thief or a finder cannot access it. iOS does not provide a remote wipe feature by default but Apple provides an application called Find My iPhone that provides this feature. This application is free and all a user must do is to create a free, so-called MobileMe, account to be able to turn on this feature. Besides remotely wiping the device, it is also possible to login and see a map and/or GPS-coordinates of where the device is located [14].

3.3 Enterprise policies

Smartphones are today often used inside companies to ease for the employees. Inside companies the IT-department administrators are often responsible for the smartphone security as well as the computer security. Usability is often more prioritized than security by users and therefore it is desired by administrators to have a way to prevent the user from using too low security on their devices. This is important in order to ensure the security of company information stored on the devices. iOS devices provide support for enterprise policies by using so called configuration profiles. A configuration profile is a file with configuration settings for a device that can be created and password protected by administrators. Configuration profiles have higher priority than user settings, which means that the user cannot change the settings and is required to follow these policies.

Configuration profiles provide much more configuration options for a device than standard settings do. A profile creator can for example force use of passcode/password, force use of stronger passwords, set the accepted time period for passwords before it must be changed and much more. Table 1 below shows a few important possible configuration categories and choices that are available with configuration profiles, (to see all possible configuration categories and choices see [15]).

Apple provides a free tool called iPhone Configuration Utility that should be used to create configuration profiles. This tool is available for both Mac and PC and can be downloaded from [16]. This tool can also be used to transfer the profile to a device and install it. When the profile is created, it can be chosen if it should be possible to remove it or not. If it is set to be removable, it can be chosen if it should require a password or not to remove the profile [15].

Table 1: Example of configuration profiles categories

Category	Description	Example of configuration choices
General	Category for all configurations for the profile itself.	Name of profile Profile identifier Profile removability
Passcode	Category for all configurations for passcode requirements.	Require passcode/password Minimum passcode/password length Maximum age of passcode/password
Restrictions	Category for all configurations for restricting the use of the device and applications.	Allow installing applications Allow use of camera Allow use of safari
Wi-Fi	Category for all configurations for Wi-Fi usage on the device.	Allowed SSID Security type on allowed Wi-Fi Password for allowed Wi-Fi
VPN	Category for all configurations for VPN settings on the device.	Connection type Server address User authentication
Email	Category for all configurations for email account on the device.	Account type Email address Use SSL

3.4 Application distribution

The only commercial way to distribute applications is through Apple's App Store. App Store is available on all iOS devices by accessing it over Internet via Wi-Fi or 3G. An application developer chooses the price for their application and Apple takes 30% of the income for their service of distributing the application on their store. To be able to submit applications for App Store, Apple requires that the developer is a registered developer and has bought a developer license from them. When buying a license Apple provides a developer certificate authorized from them. All applications should be signed with the certificate before submitting the application to App Store. This is done so that Apple can ensure that the application is coming from the correct developer and that the application has not been modified by someone else.

Before Apple accepts and makes an application available through App Store they review and evaluate the behavior of the application. This requires the developer to upload the application's binary file to Apple. Apple then checks what API:s it uses, what data it accesses and much more to ensure that applications do not do anything harmful or unaccepted. If the application passes the review, Apple signs the application with their master certificate and makes it available on App Store. Only applications that are signed with Apple's master certificate can be installed and executed on iOS devices. In this way Apple controls all applications that can be used on their devices. All modifications of existing applications, such as an update or bug fix, will require a new review before users will be able to get the change [17].

Apple also reserves the right to remove applications at any time from App Store due to exceedance of the App Store agreement. This means that they can remove any application from the store at any time if they think that it should not be there.

3.5 Baseband

The baseband is the radio module chipset together with the needed firmware for the chipset. All mobile phones have this chipset which handles all radio communication over GSM, 3G, EDGE and so on, and it is therefore a vital part for all mobile phones. The chipset is a small processor together with a modem for wireless communication. The firmware for the chipset contains among other things the instruction set for the

3 iOS platform

modem and mobile operator specific commands and data. This firmware can also lock a device to a specific mobile operator so that the device cannot be used with other operators. The current baseband used in iOS 4 devices is the X-Gold 618 chipset developed by Infineon Technologies. It has an ARM1176 processor with a clock frequency of 416MHz and a modem, which supports 3G traffic at a speed up to 7.2Mbps [18]. The baseband is loaded with its firmware at the time of manufacturing and might not be updated or changed during the lifetime of the device. In iOS devices the firmware for the baseband is digitally signed with a certificate from Apple, which should prevent the possibility to modify this firmware. Every time a device is started it checks that the baseband has the correct signature before loading the firmware.

3.6 Bootloader

Bootloading is the first step of starting up a device and it is the step where the content of the so-called bootloader is loaded into device memory and gets executed. The bootloader contains the code needed to prepare the system for starting up the operating system. The bootloader is stored in a ROM and is only used at start up. The current iOS 4 devices have actually three bootloaders that are required for starting the device and the operating system in a correct way. The first two are associated with iOS start up and they are called Low Level Bootloader (LLB) and iBoot. The remaining one is the bootloader, which loads and starts the baseband firmware (described above).

The iBoot bootloader is the one that actually initializes and starts iOS on the device. The current version for iOS 4 devices is iBoot 574.4. This bootloader checks that the operating system is a correct and valid version of iOS. If iBoot cannot validate that it is a correct version of iOS that it tries to start, it will not be started and the device will be turned off again. The iBoot itself is signed so it can be checked that iBoot is not modified to skip checks and/or load additional data into the device. LLB is checking iBoot's signature and it is more or less the only task LLB has to do together with initializing the iBoot process [19]. Figure 3 below illustrates the use of the bootloaders at the startup of an iOS device.

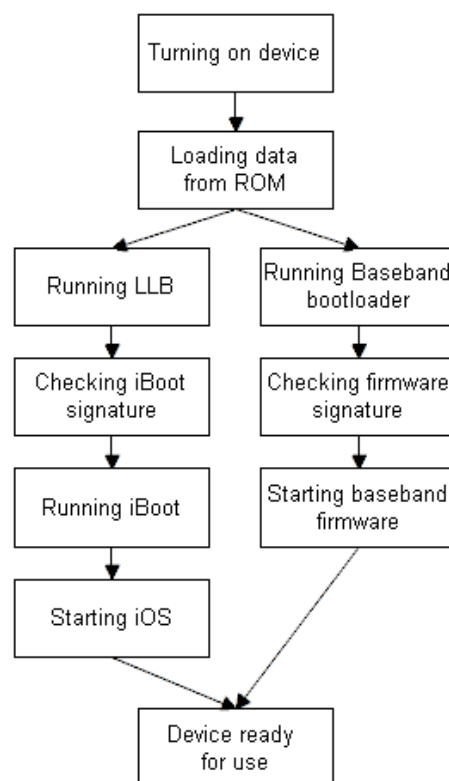


Figure 3: iOS bootloader usage

4 BlackBerry OS platform

BlackBerry OS is an operating system developed by the Canadian company Research In Motion (RIM). BlackBerry OS is specifically designed and developed for RIM:s smartphones called BlackBerrys. The first version of the operating system was released in 1999 together with RIM:s first smartphone. At the moment there is a bunch of different BlackBerry devices on the market and the latest version of the BlackBerry OS is version 6.0. There are still a lot of older versions of BlackBerrys on the market which are running older versions of the operating system and which cannot be updated to the latest version. In the beginning BlackBerrys were only targeting companies and enterprise consumers but since a few years back they are now targeting general consumers as well. Since both regular users and business users use their smartphones RIM has developed their system to be able to work in two different modes. These two modes are BlackBerry Internet Service (BIS) for regular users and BlackBerry Enterprise Solution (BES) for business users [20]. These two modes will be discussed in greater detail later in this chapter.

As with Apple's iOS BlackBerry OS's source code is not available and RIM do not comment in any greater detail on how their system works. The rest of this chapter describes the architecture, structure and security features of the BlackBerry OS platform.

4.1 Architecture

The BlackBerry platform architecture is depending on which mode the device is used in. The BIS mode and the BES mode are working in quite different ways and the platform includes different parts depending on which mode it is. (The underlying operating system has still the same architecture though). To clarify the differences between BIS and BES this chapter starts with a detailed description of these modes and their architecture. Then this chapter goes on with the operating system architecture and the hardware the system is running on. After the architecture is described this chapter continues with descriptions of the security features the system provides and information about application distribution.

4.1.1 BlackBerry Internet Service

BlackBerry devices are specifically developed to let users access Internet and emails easily wherever they are at the moment. Network access is one of the major services the device is used for regardless of regular usage or business usage. Therefore, both BIS and BES are designed to provide this service in a secure manner. BIS provides a light version of this service, which does not require any extra hardware to be used. BIS provides access to Internet through the carrier service provider's servers (or the ISP:s servers in case of using Wi-Fi) almost like any other smartphone on the market. The difference is that RIM requires that the CSP:s/ISP:s only redirect BIS traffic from BlackBerry devices to RIM's own BIS-servers. So, it is actually RIM that runs this service but from the users point of view this is not important since it only works like a regular CSP/ISP anyway [21]. Figure 4 below illustrates a simple overview of how a BlackBerry device accesses Internet using BIS.

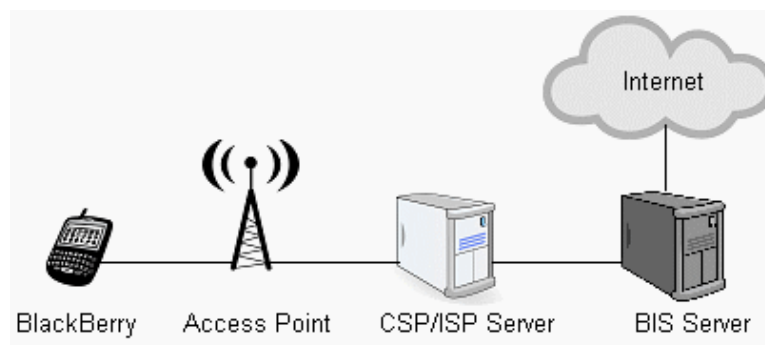


Figure 4: BlackBerry Internet Service Overview

4 BlackBerry OS platform

BIS does not provide any extra security for data traffic sent to and from the device. It uses the same security as the wireless network it is associated with uses, but it also has the possibility to use SSL if the email server/web server it connects to supports this. Many CSP/ISP servers support SSL today, and RIM's server does it, so the traffic between the device and the RIM server is in most cases protected by using SSL. BIS is mostly developed to provide an easier and less expensive way to use BlackBerrys than the more advanced version BES [22]. BIS has because of its target audience not as many extra security features implemented as BES has. So, the entire platform for BIS devices is more or less the device and its operating system without any extra hardware to count as a part of the platform.

This thesis work is focusing on the security of these platforms and the purpose of it is to come up with some suggestions on how to strengthening the security in these platforms. Organizations and governments should not use BIS since it has a lower level of security in the sense of insecure data traffic. Therefore, this thesis is only focusing on the more advanced mode BES when evaluating the BlackBerry OS platform.

4.1.2 BlackBerry Enterprise Server

BES is a more advanced and more expensive configuration mode for BlackBerry than BIS but it is also much more secure than BIS. BES provides Internet access for the device, but it can also provide a direct and secure connection to a BES server that an organization sets up and manage itself. This provides the possibility to create a connection from a device to the organization's internal network in a secure way. It makes it possible for a device to access the corporate internal servers, i.e. mail servers, web servers, application servers and so on. So, BES requires that the organization have their own BES server managed in a secure way and stored in a secure location inside the company's firewalls [20].

When using BES mode the device will encrypt all data traffic using symmetric key encryption at the transport layer. The data stays encrypted all the way to the BES server, which is the only one that can decrypt it. When the BES server sends data back to the device it also encrypts all data traffic in the same way. In this way no data traffic is ever left unprotected, which prevents third parties such as a CSP/ISP to be able to access the organization's private information. Figure 5 below illustrates a simplification of how communication between a BlackBerry device and a corporate internal network is working.

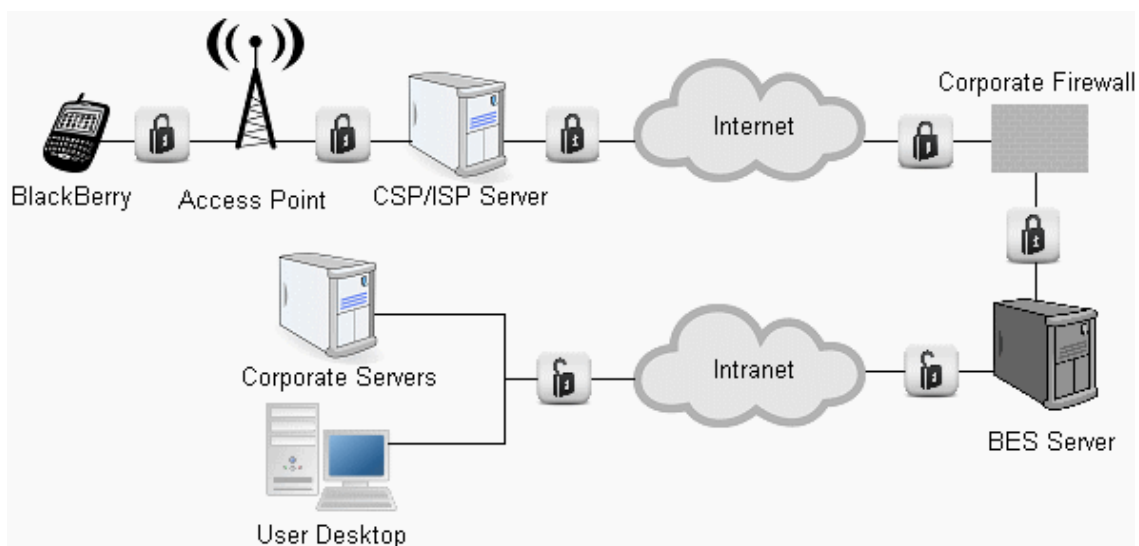


Figure 5: BlackBerry Enterprise Solution Overview

The BES server also integrates with enterprise messaging systems to provide email access and instant messaging for a device. With the current version of the platform BES has support for use of Microsoft Exchange, IBM Lotus Domino and Novell GroupWise messaging systems [22]. The TCP/IP connection that is opened between a device and a BES server is outbound initiated from the BES and the connection is only

opened to authenticated devices. Once the connection has been established it is kept open at all times. This makes it safe to open the required port (3101) in the company firewall since connections only can be established from inside the firewall and are then kept open.

BES mode provides a bunch of security features that makes the BlackBerry OS platform quite secure when using this mode. A detailed description about the extra security features that are implemented in this platform is given in section 4.2 later in this chapter.

4.1.3 Device architecture

The later versions of BlackBerry OS are based on Java and all applications for this operating system must be written in Java. The operating system can be divided into four layers, which each has their own important role in the system. Figure 6 below illustrates an overview of these layers. The layers from high-level to low-level are: application layer, runtime layer, OS kernel and hardware layer. The top layer, application layer, is representing all running applications in the system. This includes third party applications as well as system applications such as web-browser, email client, games and so on. The next layer is the runtime layer and it is representing the runtime environment, which all applications are executed within. The current version of the operating system is using a Java 2 Platform Micro Edition (J2ME) as runtime environment. J2ME is a platform designed specifically for mobile and embedded systems. Included in the J2ME environment is a Java Virtual Machine (JVM) used for optimizing execution on devices. This JVM is designed and developed by RIM to match the exact requirements for their devices and it is called BlackBerry JVM. RIM has left out some common Java functionalities in their JVM on purpose to make the system less modifiable for the users. For example, JNI and Java reflection are not supported in the BlackBerry JVM. Besides the JVM this layer is also including a bunch of J2ME API:s that can be used in applications to access hardware and kernel resources at runtime.

The third layer from the top is the operating system kernel layer and it represents the kernel of the operating system. BlackBerry OS's kernel is Java-based to provide required low-level functionality for the runtime environment [23]. Just like in iOS, the kernel is handling threads, low-level network structure, I/O system and the file system. It also includes the drivers needed for the operating system to be able to use the underlying hardware.

The bottom layer of the architecture model is representing the underlying hardware of the system. There are a few available different devices on the market which all can run the latest version of BlackBerry OS and these devices have some differences in the hardware. The common parts of the hardware for all devices running BlackBerry OS 6 is that they are always running on a processor built on the ARM architecture, (depending on the device it uses ARMv7 or ARMv9). Most BlackBerry devices use a processor from Intel and the clock frequency for the processor is between 500-900MHz depending on the device [24].

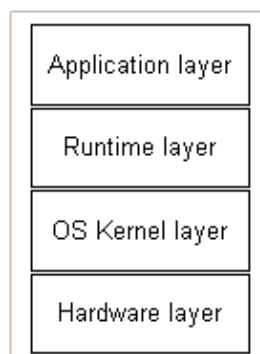


Figure 6: Blackberry OS layers

4.2 Security features

BlackBerry OS provides many extra security features in order to make the system much more secure and safe to use. Especially devices used in BES mode have a great amount of extra security features available. Features like device encryption, user authentication and message encryption are provided and in this section these security features will be described and discussed. As stated above this thesis work is only focusing on BlackBerry OS used in the more secure business mode, BES, and does not discuss BIS security at all.

4.2.1 User authentication

There are a few different authentication methods available on a BlackBerry. The default version is using a user chosen password to authenticate the user to the device. BlackBerry recommends by default a password with both characters and digits. That is the default version, but the operating system also provides the possibility to use versions of two-factor authentication on the device. This means that not only a password is required to unlock a device; it also requires another type of authentication. The default available versions are using a smart card or memory card containing a secret key that must be inserted into the devices together with a password to unlock the device [22]. There exists also third party developed authentication methods that can be integrated into the system. Methods such as touching the screen in a secret pattern and biometrics using for example face recognition by using the built in camera is available by third party applications, but these are not available in the system by default. The two-factor authentication is more secure than only using a password since it combines the requirements of knowing the user's secret password and also having access to the user's token.

4.2.2 Data encryption

BlackBerry devices have built in support to encrypt all data that is stored on the device while the device is locked. All the user has to do is to choose a secure password for the device (or use a two-factor authentication as described above) and turn on the feature called data protection. BlackBerrys are equipped with hardware for both AES encryption using 256-bit keys and Triple-DES using 112-bit keys. The BlackBerry OS is designed to encrypt all data that is stored on the device with the AES algorithm when the device locks [22]. All data that is received at the device while it is locked, e.g. email and SMS, is also being protected using encryption but instead of using AES the device uses the supported Elliptic Curve Cryptographic algorithm (ECC) and uses the public ECC encryption key. The length of the ECC keys that is used in the device is decided by the user and should be carefully considered to be large enough for the wanted level of security. To achieve equivalent level of security with ECC as with AES using 256-bit keys it is recommended to use ECC keys with a size of 521-bit [25]. When decrypting the data the same AES key is used for the stored data and the private ECC key is used to decrypt the received data.

The AES encryption key for this action is called the content protection key and the ECC keys are called the ECC encryption key pair. All these keys are by default generated at the device at the first setup. These keys are generated without any user input and the user has no knowledge about them. So, the user chosen password is not used to generate these keys at all, instead it is used to generate an ephemeral key. The ephemeral key is used to encrypt the content protection key and the ECC keys when they are stored on the device [22]. Section 4.3 in this chapter describes in greater detail the different keys in the system, how they are generated, what they are used for and how they are stored on the device.

4.2.3 Communication security

As described earlier in this paper BES is founded on secure data communication between the device and the BES server. This is achieved by using transport layer encryption for all data communication between the two parts. The used encryption algorithm depends on the configuration of the BES server. A BES server is required to at least support Triple-DES but is recommended to support AES as well. A BES server is also required to always use the most secure algorithm that is supported by both the device and the server [22]. If the AES algorithm is used the system generates a 256-bit message key and if the Triple-DES algorithm is

used the system generates a 112-bit message key.

Besides the default encryption of all transferred data BlackBerry OS provides support for both SSL/TLS and VPN as well. SSL/TLS is by default used whenever it is possible but VPN must be configured and activated by the user to be used.

In BlackBerry OS it is possible to send so-called PIN messages, which are a type of message that is sent between BlackBerry devices directly and does not pass through the BES server at all. Each BlackBerry has a unique PIN, which acts like the address for this type of message. Emails and PIN messages are by default protected with both the communication protection and with message integrity protection that prevents that messages are being modified on the way between a device and the BES server. This is done by using generated message keys. So, messages in the system are by default quite well protected, but if a user wants to extend this security even more PGP encryption is supported for emails and PIN messages. This means that the message stays encrypted with the PGP keys all the way from the device to the receiver and is not decrypted when arriving at the BES server [22]. BlackBerry OS usually generates all used encryption keys, but in the case of using PGP the user must generate the keys himself and then manually place them on the device. BlackBerrys support PGP encryption using the AES algorithm with 128, 192 or 256-bit keys, the CAST5 algorithm with 128-bit keys or Triple-DES using 168-bit keys [22].

4.2.4 Sandbox

Just as with iOS, BlackBerry OS also using a kind of sandbox to encapsulate all running processes. Each process is encapsulated in their own part of the memory with restricted permissions and access rights. BlackBerry OS only lets an executing application access files and data that are stored in the application's directory. This prevents processes from being able to access and manage sensitive information, which they should not be able to access. In this way it will also effectively prevent a possible malware to spread over the entire system.

The only way to access resources and files outside the application directory is by using one of the system-provided API:s. Such API:s can grant access to network sockets, cryptographic functions, system data and much more, but it is highly controlled what can be accessed by the API:s so it cannot be exploited in a non-acceptable way [22].

4.2.5 Memory clean and device wipe

Sensitive data is often loaded into different parts of the memory when used and therefore are BlackBerrys designed to securely clean the memory to ensure that data is not left accessible in the memory. This behavior is turned on by default in BlackBerry OS and it cannot be turned off if data protection is activated. The device is by default set to clean the memory whenever it can, but the user can configure this. The user can among other things choose to clean the memory when a device locks, when a device synchronizes with a computer or when the device has been idle for a certain time [22].

A user can decide to wipe all data on a device as well. This means that all data on the device is deleted in a secure way. It can be done directly if a user feels that he cannot guarantee the safety of the device, or it can be configured to wipe all data if certain actions occurs, e.g. if the devices is locked for a certain time without being unlocked, if the device battery is running low or when the device receives a remote wipe command from the BES server [22].

A device can also be configured to wipe all data on the device if a user enters the wrong password too many times in a row. The number of allowed attempts can be set by the user and can be chosen to be between three and ten attempts.

4.3 Key management

The security of BlackBerrys using BES is much about encryption. Encryption is heavily used and is used for

4 BlackBerry OS platform

a lot of different things in the system. Therefore, lots of different encryption keys are required to achieve secure encryption. Most of the used encryption keys are generated by the system without any user input and the user does not have any knowledge about the key itself. BlackBerry OS uses at least eight different encryption keys to encrypt the different types of data that a device handles. Table 2 below gives an overview of the existing keys, their size and what they are used for [22].

Table 2: Encryption keys on a BlackBerry device

Key	Algorithm	Key size	Description
Content protection key	AES	256-bit	Encrypts the device data and the Principal encryption key.
Device transport key	AES/Triple-DES	256/112-bit	Encrypts the Message keys.
ECC private key	ECC	Optional	Decrypts the data that is received at the device when the device is locked.
ECC public key	ECC	Optional	Encrypts the data that is received at the device when the device is locked.
Ephemeral key	AES	256-bit	Encrypts the ECC private key and the Content protection key.
Message keys	AES/Triple-DES	256/112-bit	Encrypts the data that is transferred to and from the device.
PIN encryption key	Triple-DES	112-bit	Encrypts PIN messages send to and from the device.
Principal encryption key	AES	256-bit	Encrypts the PIN encryption key and the Device transport key.

Some keys in the system are used to encrypt other system keys so that keys are not stored in plaintext on a device. This creates a key hierarchy where the lower keys are protected by the higher keys and so on. This goes on up to the ephemeral key which is highest in the hierarchy. Appendix B: BlackBerry encryption key hierarchy shows an overview of the key hierarchy in BlackBerry OS in order to clarify how keys are protected on the device.

The ephemeral key is not stored on the device at all. This key is instead generated from the device password each time it is needed. So, the ephemeral key is only valid as long as the user keeps the same password on the device. When a user changes the device password, a new ephemeral key will be generated and all data that is encrypted with the old key will be decrypted and then encrypted again with the new ephemeral key. The generation of the ephemeral key is done by using the PKCS#5 protocol. The ephemeral key is used for encrypting the ECC private key and the Content protection key when those are stored on the device [22].

The Content protection key and the Principal encryption key are generated when the data protection feature is turned on for the first time. These keys are generated by using a pseudorandom number generator and are then stored encrypted in the flash memory of the device. The Content protection key is used to encrypt all data that is stored on the device when the device is locked. It is also used to encrypt the Principal encryption key when this is stored on the device. The Principal encryption key is used to encrypt the Device transport key when this is stored on the device.

The ECC key pair (public and private key) is also generated when the data protection feature is turned on for the first time. The public key is used to encrypt all data that the device receives while it is locked. The private key is then used when the data should be decrypted. The key pair is stored in the flash memory of the device and the private key is stored encrypted while the public key is stored in plaintext [22].

The first Device transport key is generated when the device is activated at the BES server for the first time. It is the BES server or the BlackBerry desktop software that generates the key and transfers it in a secure way

to the device. This key will be renewed at the BES server after a certain time (by default 30 days) and transferred to the device encrypted using a message key. In this way the system will never use the same transport key for a too long time. The Device transport key is used to encrypt Message keys, both when they are stored on the device and when the BES server and the device exchange Message keys. This key is also used for encrypting the stored PIN encryption key.

Message keys are randomly generated at the BES server or at the device when a part wants to send data to the other part. All data that is transferred between them is encrypted with a Message key and the Message key itself is exchanged between the parts encrypted with the Device transport key. A unique Message key is used for all data that is larger than 2KB and that is divided into several packets. The last key in the system is the PIN encryption key and it is used for encrypting PIN messages sent between BlackBerry devices. This key is by default a global key so it is the same for all BlackBerry devices. However, a company can setup a unique PIN encryption key so that only company employees can decrypt the PIN messages sent from company BlackBerrys [22].

Most of the used encryption keys are generated on the device automatically without any user interaction. Many of these keys are generated by using the DSA pseudorandom number generator. The DSA PRNG was developed to be the standard for generating pseudorandom input for the DSA signature algorithm. In BlackBerry OS DSA PRNG is based on the hash-function SHA-1 and it can take an optional input as an extra seed for the number generation [26]. The Content protection key and the Principal encryption key are generated with the DSA pseudorandom number generator without the extra input. For each of the keys is a pregenerated secret stored on the device and this secret is aggregated with the current timestamp using the XOR operation. The result from the XOR operation is then hashed with SHA-512 before it is used as a seed for the pseudorandom number generator.

The first time a Device transport key is created it gathers cursor coordinates from the user for a total of 384 bytes. Then is 384 bytes taken from the Microsoft Cryptographic API and these 768 bytes are hashed with SHA-512 to produce 64 bytes of data. From those 64 bytes are the first 256/112 bits taken and those are used as the encryption key. For later key generation the generated 64 bytes are used as a seed for the SHA-512 hashing and this generates the new keys [22]. The Message keys are also generated on the device by using DSA PRNG with a seed. The seed is generated from a pregenerated secret that is aggregated with the timestamp using XOR operation and then hashed with SHA-512. The Message key generation differs from the other keys in the sense that it is using 64 bytes of extra input. The input is generated by taking data stored on the device and using the ARC4 algorithm to scramble the data and from that take 521 bytes of data. This data is then hashed with SHA-512 to generate 64 bytes, which are used as an extra input to DSA PRNG [22].

4.4 IT-policies

The BES mode is specifically designed to be used by organizations that want to achieve a secure system for their devices. Such organizations often have their own policies for how their data and systems should be used to achieve the right level of security for their organization. Therefore, companies have administrators that supervise their systems and manage the security of the system. BlackBerry OS together with BES mode provides tools for administrators in order to set up BlackBerrys so that the devices must comply with the security requirements for the company's systems. These tools make it possible for the administrators to setup IT-policies on the BES server, assign devices to certain policy groups and then push the policy restrictions to the devices automatically from the BES server [27].

There are dozens of different policy groups with a total of hundreds of possible configuration restrictions for BlackBerrys. This makes it possible for administrators to almost exactly manage devices in a way suitable for their companies. Everything from password requirements, encryption management, Bluetooth usage and application restrictions are possible to configure with these policies. For a total listing and description on available IT-policies see [27].

4.5 Application distribution

Unlike iOS there are many ways to distribute applications for BlackBerry OS devices. That is because it is not required that all applications are signed by RIM to be able to run in BlackBerry OS. Since there is no requirement that RIM must evaluate and sign applications they can be distributed in pretty much any way, e.g. at an download from internet, an email attachment, from memory cards and so on. Hence, it is up to the users to choose if the application source can be trusted or not. RIM nevertheless provides a store for applications that is called BlackBerry App World. On App World are only applications that are evaluated and tested by RIM available. RIM does not take any of the income from applications distributed through BlackBerry App World, but they take out a fee of \$200 for every ten applications a developer uploads to the App World [28]. There is third party application stores on the Internet for BlackBerry applications, which do not require fees for distributing applications and do not require evaluation of applications. One of the most popular third party stores is MobiHand [29], which was available even before BlackBerry App World existed.

No evaluation and no signing is required to install and distribute applications in BlackBerry OS, nevertheless the system has other ways for preventing applications to do malicious things. As stated earlier the system is using a kind of sandbox model together with the JVM to prevent applications from accessing resources directly. It requires applications to use provided API:s to do so. To raise the security even further BlackBerry OS classifies its API:s in three classes and restricts the use of them. The API classes that exist are the public class, the private class and the protected class. The public class contains all non-sensitive API:s that do not provide any access to user or system sensitive resources. These API:s can be used by any application without any restriction. The private class contains all highly sensitive API:s that provide access to system sensitive resources. Private API:s may not be used in any third party applications. The JVM prevents applications that try to access these private API:s from being executed in the system [22].

The protected class contains all API:s that provide access to user and system sensitive resources that might be needed in applications. Therefore, third party applications may use these API:s to access sensitive resources. However, to be able to execute applications that use protected API:s the application must be signed by the developer. To get the needed certificate for such a signature the developer must register at RIM that his application needs to access protected API:s. The developer must also motivate why the application needs to access these API:s. If RIM accepts the motivation they will provide the developer with a code signature certificate to use for signing the application. In this way does RIM keep track of what applications that access sensitive data and who the developer actually is [30].

4.6 Baseband

The baseband chipset differs among the available BlackBerry devices but in the later versions are most of them using some chipset from Qualcomm with a clock frequency around 600MHz. A common baseband processor in BlackBerry devices is the Qualcomm MSM6700. All BlackBerry devices use a baseband chipset that at least supports GSM, GPRS, 3G and Wi-Fi [24]. Most of the later models have a chipset that supports GPS as well.

Just as for iOS the BlackBerry baseband firmware is loaded into the device at the time of manufacturing. The baseband will generally not be changed during the lifetime of the device. RIM digitally signs the baseband firmware so that the device can perform a check at boot time to see that the firmware is correct and not modified by any third party.

4.7 Bootloader

The bootloader is loaded into memory at boot time of the device. The bootloader is the piece of code that loads and starts the operating system and its components on the device. All BlackBerry devices have two bootloaders, one for the BlackBerry OS and one for the baseband firmware. These bootloaders are only used at the time of device start up. To protect the integrity of the operating system the system is digitally signed

and the bootloader checks that this signature is valid and correct at boot time. If the operating system does not have a valid signature the bootloader will turn the device off again.

The bootloader is also digitally signed to ensure that the bootloader itself is not modified. The bootloader signature is checked by the processor at start time and if the signature is invalid the bootloader is not allowed to start. Hence, the device will not start if the signature check fails [22]. The baseband bootloader is working in a similar way and it checks the signature of the baseband firmware before allowing the baseband module to start.

5 Proof-of-concept spyware

It is often said that iOS is secure and that it is well protected against malware. Their sandbox model is said to stop any application from accessing sensitive information on the device, and all applications must be verified by Apple before they can be installed on a device. All these things are said to make the system safe from malware. So, is iOS totally protected against malware? Absolutely not and therefore has a proof-of-concept spyware been developed in this thesis to point out the risk of malware. The purpose of the application was to see what information on the device that can be extracted from an application without notifying the user about it. This application does not use any private or forbidden API:s and it does not require any user interaction to work besides that it is installed and started on the device. The information that is accessed inside the application is accessed in the background of the running application and it should not be visible for the user that this information has been accessed. However, this is only a proof-of-concept application and therefore no effort has been made to create a real application to hide the malicious code in.

Besides checking what data that can be accessed this application has been implemented to upload all data to a FTP-server. This is done in order to show that sensitive data can be transferred from inside an application. This application has been tested on an iPhone 4 running iOS 4.3.3 but in theory a similar spyware can be created for BlackBerrys as well. However, this has not been tested in this thesis work.

5.1 Accessible data

There are a lot of data in an iOS device that can be accessed from inside an application. This application gets access to some very sensitive data without using any private API:s and without notifying the user about it. This application has eight different categories for information that can be accessed and these are: system information, address book, keyboard cache, locations, dialed numbers, email settings, Wi-Fi networks and microphone recording. Below each category and its information are briefly explained.

System information: Contains information about the system, which the application is running in (in this case iOS version 4.3.3). The unique device serial is accessible as well as the name of the device. The more sensitive information in this category is the current IP-address and the device MAC-address that can be used for further attacks. Even information about the carrier and the SIM-card's unique id number (ICCID) is accessible.

Address book: The entire address book with all contacts and information is accessible within this application. This information can be very sensitive for a user and if it ends up in the hands of an attacker surveillance of the user and/or a company will be much easier. Worth noting is that it is not only possible to copy this information, it would be perfectly possible to delete, add or modify contacts as well. This might in some cases be even worse than that an attacker accesses the contacts.

Keyboard cache: iOS devices have a cache file containing words typed on the keyboard and this cache is not possible to clean from the device. In this cache file is almost everything ever typed on the keyboard saved. (It does not save passwords or words that are shorter than three characters). This cache is used in order to help the user to auto-correct misspelled words and sentences that the user frequently writes. This information is of course sensitive even though passwords are not saved, since emails and messages pretty much can be read in clear.

Locations: iOS devices are equipped with GPS and a lot of applications are trying to use this to establish the current location of the device. However, when an application tries to access the GPS-module it is required to ask the user if he allows this. This application can of course access the GPS-module directly but to avoid notifying the user that the application wants to know the location, this application instead accesses stored GPS-coordinates of previously visited places. A lot of applications, e.g. Maps, are storing the latest known location and this information is accessible for this spyware. The coordinates together with the time of visiting are sensitive information, especially for users that frequently uses Maps (and similar applications).

Dialed numbers: This spyware is looking up the last number that is dialed on the keypad as well as the last

dialed contact. It should be possible to extend this to include all dialed numbers and contacts in the call-list.

Email settings: Most iOS users are configuring their device to be able to get their emails to their phone. Settings for all email-accounts that are configured on the device are accessible for this application. Information such as full email-address, username, hostname (e.g. imap.gmail.com for Gmail-accounts) and account-id is accessible. This might not be the most sensitive information to leak for the average user but it can still be sensitive for some users.

Wi-Fi networks: A list of all Wi-Fi networks that are saved in the device's known Wi-Fi-list is accessible. This includes the name of the network (SSID), the MAC-address of the access point, timestamp of first joining and timestamp of the latest usage. This can be very sensitive information since the SSID can be used to trick the device to connect to a rogue Wi-Fi access point and in this way make further attacks.

Microphone recording: The device is equipped with a microphone, which can be turned on in order to record the sounds from the surroundings of the device, and this is done by this application. The microphone is turned on and records every sound around the device without notifying the user at all. This is of course sensitive, since conversations close to the device as well as phone calls can be recorded and uploaded to the server. This is probably the most frightening part of this application, since it so clearly violates the user's privacy.

5.2 Behavior obfuscating

Since this application is performing actions that are not allowed according to Apple's user agreements it is a good idea to obfuscate the behavior of the application. Therefore, some features have been implemented in this application to try to hide what the application does. Both the resource access and the data that is uploaded have been obscured to make it harder to see what this application really does. To access resources from within an application the information must be loaded from a file into some data container. Figure 7 below is showing some source code from this application, which loads the keyboard cache file into the application.

```
// Load the keyboard cache into an array with words
-(NSMutableArray *)spyKeyboardCache {
    NSString *file = @"/private/var/mobile/Library/Keyboard/sv_SE-dynamic-text.dat"; // Swedish words
    NSString *fileContents = [NSString stringWithContentsOfFile:file encoding:NSUTF8StringEncoding error:nil];
    NSArray *wordArray = [fileContents componentsSeparatedByCharactersInSet:[NSCharacterSet controlCharacterSet]];
    ...
}
```

Figure 7: Code for accessing keyboard cache

As seen in Figure 7 the file is identified with the full path of the file. By looking on these strings it is easy for anyone to see what resources this application accesses. This is for instance one of the things Apple's reviews are said to automatically check. To prevent that this information can be automatically checked string obfuscating has been implemented in this application. Figure 8 below shows the same piece of code but with the obfuscating action implemented.

```
// Load the keyboard cache into an array with words
-(NSMutableArray *)spyKeyboardCache {
    NSString *file = [self decipherString:@"SXdBmTEqSmTdSDfCBPqSOBCdTdZSxqZCfTdoSrmsH.coZvTDBicEqkEMoTE"];
    NSString *fileContents = [NSString stringWithContentsOfFile:file encoding:NSUTF8StringEncoding error:nil];
    NSArray *wordArray = [fileContents componentsSeparatedByCharactersInSet:[NSCharacterSet controlCharacterSet]];
    ...
}
```

Figure 8: Code for accessing keyboard cache with string obfuscating

The function decipherString in Figure 8 is just a small function that uses a simple substitution cipher for the input string and returns the string in plaintext. Of course it is totally possible to figure this out and decipher

the string to see what the application accesses, but this obfuscating does prevent automatic checks to react on the behavior [31]. After all, it is not possible for Apple to review all applications manually since the amount of applications is huge.

To hide what data that is uploaded to the FTP-server is SSL used. The application is using SSL so that data is sent encrypted. This is after all a normal behavior for an application since many applications have features like data backup to servers. Hence, SSL usage is not considered suspicious in most cases. However, the use of SSL might require an application to make special efforts to be accepted to App Store according to USA's rules about encryption usage, i.e. it needs permission for encryption usage.

These above mentioned actions are simple but it still makes it much harder to find out that an application behaves suspiciously. Since there are no detailed descriptions about how Apple performs their review it is not possible to tell if these actions are enough to make the application pass the review or not. However, for jailbroken devices it is possible to bypass the review and with physical access to a device the device can easily be jailbroken.

Furthermore, this application is not embedded in some other application since it is only a proof-of-concept application. In the real case the malicious code would be imbedded in some other application, such as a game, which makes it much less obvious what the application's intention is, and it would make it much more likely that the user will execute the spyware code.

5.3 Conclusions

In only a few hours was a spyware application that accesses and transfers a great amount of sensitive information implemented without using any private API:s. Actions to make this application pass Apple's application review has been implemented and it is reasonably possible that such an application passes the review and becomes available on App Store. Even if this spyware might be discovered quite fast and then removed from App Store it can still have infected a great amount of users before it is detected. As stated earlier it should be equally possible to implement such a spyware for BlackBerry, and it might even be more effective since it is easier to distribute applications for BlackBerrys, but this has not been tested to any greater extent.

There exist a lot of sources on Internet with information about what can be accessed, how it can be accessed and how to pass application reviews. Both descriptions and source code are available which makes it even easier to implement this kind of applications. Pretty much anyone with some experience in programming can do it. For instance was [31] used as inspiration and for information about how to implement this proof-of-concept spyware.

The ease of making this kind of malicious code together with the high level of sensitivity of the accessed data makes malware a threat that smartphone users indeed should worry about and try to protect their devices from. In Figure 9-Figure 11 below are some example images of this proof-of-concept application shown.

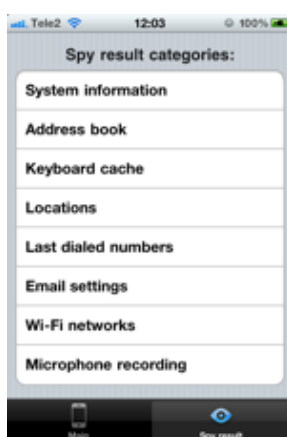


Figure 9: Spy categories



Figure 10: System information



Figure 11: Keyboard cache

6 Rogue GSM base station

GSM is an abbreviation for Global System for Mobile communication and it is still the largest communication network for mobile communication in the world. More than 80% of the world's population has coverage for GSM network and it is by far the most used mobile communication system [32]. GSM has been up and working since the beginning of the 1990's. Although newer systems, such as 3G, are used more and more, GSM is still really important for the world wide mobile communication.

GSM is known to contain a few architectural flaws that make the system insecure to use. For instance some of the encryption algorithms used by the system have been proved to be insecure and can be cracked in a short time. The problem with weak encryption algorithms has been known for quite some time but there exists other threats for the security of GSM that recently has become a more feasible threat. A phone that is connected, without the users knowledge, to a rogue GSM base station provides the possibility of a man-in-the-middle position for the attacker. This kind of attack has always been possible but in earlier days the price for setting up such a base station was really expensive and it used hardware that was really hard to get, but today this hardware is available for anyone to a reasonable price. The availability and the low cost make this threat much more probable to happen.

Both Apple's iPhone and RIM's BlackBerrys is developed to be able to use the GSM network and are therefore possible victims for this kind of attack. This thesis has therefore included some practical testing with the aim to set up a rogue base station and investigate what can be done and how severe it would be for the victim that encounters the attack. This chapter presents the result and conclusions of this investigation of GSM attacks.

6.1 Attack description

The attack that has been investigated in this thesis is a kind of man-in-the-middle attack for mobile phones using the GSM network. The attack is about setting up a rogue base transceiver station, (also known as a rogue BTS), to trick victim phones to connect to it and in this way let the attacker eavesdrop on all calls made and read all SMS that is sent. From the victim's point of view everything will seem normal in the sense of that the victim can still make calls and send SMS as usual. It is also possible for the attacker to make phone calls and send SMS to the victim with a spoofed phone number, so the attacker can pretend to be whoever he wants to be while calling and sending SMS.

Another important part of the attack is the possibility of creating and sending so called over-the-air programming messages from the base station to phones connected to it. Over-the-air-programming messages, also called OTAP messages, are a kind of message that distributes updates of phone software and phone settings [33]. These messages can for example change the APN-settings on a device or update the firmware of the baseband module. OTAP messages can in theory be misused by the attacker to infect the victim phones with malware and other malicious things.

The above described is the aim of this attack and the following sections are describing in greater detail how and why the attack works.

6.1.1 GSM limitations

The above described attack works due to some limitations in the GSM architecture combined with some restrictions from the mobile phone manufacturer. First of all the GSM system is designed in such a way that base stations never authenticate themselves for a phone that wants to be connected; only the phone is required to authenticate itself for the base station. This limitation leads to the possibility to trick a phone that the rogue BTS is a real and trustable base station. The second limitation of the GSM system, which makes this attack possible, is that it is always the base station that decides which level of encryption that should be used between the phone and station. This leads to the possibility of letting the rogue base station use the encryption level A5/0, which means no encryption at all. The good thing with GSM in this case is that it does

6 Rogue GSM base station

notify the phone that no encryption is used, so a victim should be able to note this, but here comes the first restriction from the phone manufacturers. Almost no phone models on the market today do notify the user about this, the warning sent by the GSM network is in almost all cases just ignored since it is said to be confusing for the user.

Another restriction is that almost all phones on the market are designed to fall back to GSM whenever the more secure 3G connection is low and/or lost. This is done so that a user should be able to make calls even though the more limited 3G coverage is out of reach. It is also common for today's phones to always connect to the GSM-network that currently provides the strongest signal power [34].

So, to summarize; phones always connect to the GSM base station with strongest signal power, the base station is not authenticating itself for the phone and the user is not notified about that his calls and messages are sent unencrypted.

6.1.2 Attack setup

The setup of this attack is, as mentioned earlier in this chapter, to make victim's phones connect to a rogue base station. Today's mobile phones connect to the GSM base station with the strongest signal power so the rogue base station should be placed close enough to the victim so that this base station has stronger signal power than the real surrounding base stations. The rogue base station is configured to use encryption A5/0 that means no encryption at all. The base station is connected to a laptop that is configured as a GSM server and as a software phone switch. When the victim makes a call the laptop will process it and send it as a SIP-call over the internet to an external SIP-gateway, which transfers the SIP-call to the telephone network (PSTN) and connects the call to the callee. In that way is the victim's call established and the victim is not notified that his call can be eavesdropped on and recorded at the laptop. Figure 12 below illustrates a simplified overview of a phone call in the attack scenario.

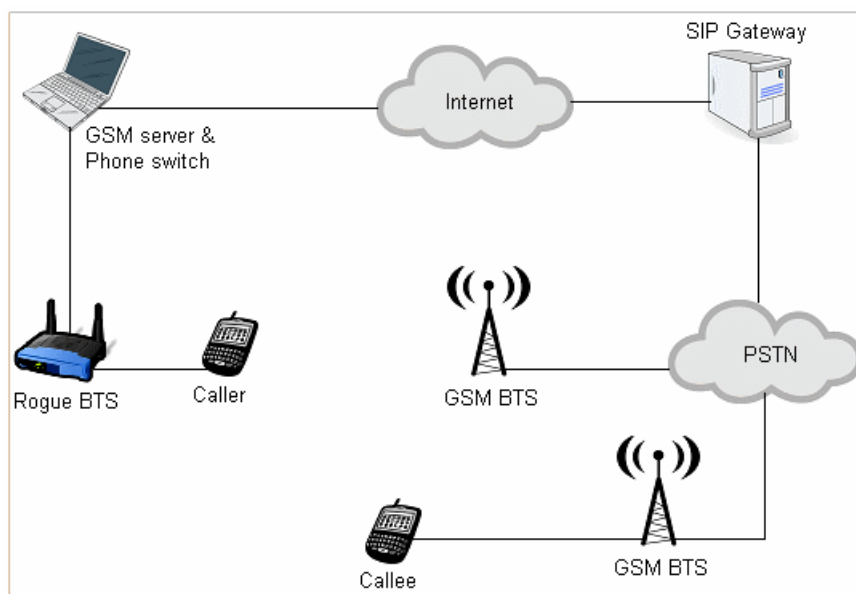


Figure 12: Overview of the attack scenario

6.1.3 Legal restrictions

In Sweden, as well as many other countries, the permission to transmit in the GSM frequencies is limited by law. So, to be able to set up the rogue base station and perform this attack analysis, permission from the country's authority of radio transmission is required. In Sweden, this authority is "Post- och telestyrelsen" [35], and they have granted Sectra Communications permission for this testing in the frequency 1805.2MHz.

and the absolute radio-frequency channel number 512. Therefore, has all testing for this attack been performed at this frequency, but the attack is possible to perform at any GSM frequency. All testing has been made in a shielded lab according to the Swedish law.

The legal restriction is of course only a matter for testing; in the case of a real attack on a real victim the attacker would probably not care about the transmission law, since he will break the law by eavesdropping on the victim anyway.

6.2 Attack equipment

The equipment needed to set up a BTS used to be really expensive and it was very complex to implement it until a few years ago, when an open source project named OpenBTS was released. OpenBTS is the software needed for setting up a GSM server with a GSM stack, and this project led to the development of a small Universal Software Radio Peripheral to use as hardware for the BTS. Hence, suddenly there existed an open source project and relatively cheap hardware for a BTS on the public market. This piece of hardware and the open source project have been used to set up the rogue base station for this test. In the following sections is the used equipment described in greater detail.

6.2.1 BTS hardware

All the BTS hardware for this test attack has been bought from Ettus Research LLC, [36]. All the needed hardware for this BTS was bought for \$1470 (~9000SEK) including shipping cost from USA to Sweden. The hardware used in this attack is listed in Table 3 below.

Table 3: List of used hardware for rogue BTS

Amount	Name	Description	Cost
1x	USRP1 - Kit	Universal Software Radio Peripheral including motherboard	\$700
2x	RFX1800 Daughterboards	Transceiver cards (can transmit and receive) at 1.5GHz-2.1GHz	\$275 (each)
2x	VERT900 Antenna	Antennas for the daughterboards	\$35 (each)

The USRP1 is a Universal Software Radio Peripheral type 1 and this kit includes: steel box, motherboard, fan, USB-cable, power adapter, two SMA-cables and all screws needed to assembly the USRP. The RFX1800 daughterboards are the transceiver cards for the radio module. They can both transmit and receive radio signals in the frequencies 1.5GHz-2.1GHz. One transceiver card is used for the downlink and one for the uplink. Each daughterboard has a VERT900 antenna connected to be able to transmit and receive. This is all hardware needed for setting up a small GSM BTS.

6.2.2 BTS software

All software needed for this testing is based on open source projects and is therefore free of charge. All softwares are developed for Linux distribution and it is recommended to use either Ubuntu or Debian. Therefore, Ubuntu version 10.10 Maverick Meerkat has been used as platform for the server. Ubuntu can be downloaded at [37]. The USRP also needs some drivers (UHD) to work properly, and these can be downloaded at Ettus homepage [36], for this testing has the UHD version 003.000.001 for Ubuntu been used. An overview of all software used for the base station is listed in Table 4.

An open source project called GNU Radio has been used as well. GNU Radio is a software implemented to provide signal processing at a low-cost for implementation of software radios. The GNU Radio project's main purpose was to ease for non-professional and academic research in the field of wireless radio communication. The latest version at the moment of writing is GNU Radio 3.3.0 but for this testing has the

6 Rogue GSM base station

older version GNU Radio 3.2.2 been used since the latest version showed unstable behavior when used together with OpenBTS. This is not common, since the latest version is tested and said to work properly with OpenBTS, but for this test it did not work as wanted. Anyway, both versions can be downloaded at [38].

OpenBTS, as mentioned earlier, is a software that provides the air interface, stack and server for a GSM network. It requires GNU Radio and Asterisk PBX to work properly. GNU Radio is needed for OpenBTS to be able to control the USRP and Asterisk PBX is a software telephone switch that processes phone calls made from within the network. OpenBTS processes all GSM signals and redirect calls to the phone switch in the format of SIP-calls. To provide the possibility to send SMS messages in the GSM-network OpenBTS uses another software called Smqueue. Smqueue has been integrated in the later releases of OpenBTS. For this testing has the latest version of OpenBTS been used which is OpenBTS 2.6.0 Mamou and it can be downloaded at [39].

Table 4: List of used software for rogue BTS

Software	Version	Description
Ubuntu	10.10	Operating system for the server
Universal Hardware Driver	003.000.001	Drivers for the USRP
GNU Radio	3.2.2	Software providing signal processing for software radios
OpenBTS	2.6.0	Software providing GSM-stack, server and air interface
Asterisk PBX	1.6.2.7	Software implementation of telephone switch

Asterisk PBX is an open source software implementation of a telephone switch, which redirects phone calls between connected phones and communication networks. Asterisk is used to connect the calls that the GSM interface is picking up and sending to Asterisk. In this testing has Asterisk been configured to send the calls as SIP-calls to a SIP-client or to a SIP-gateway that redirects the SIP-call as regular phone calls into the public telephone network. Asterisk is also able to connect calls between mobile phones connected to the rogue base station. The latest version of Asterisk PBX is at the moment of writing 1.8.3.2, but it is recommended to use Asterisk PBX version 1.6.2.7 together with OpenBTS and therefore has the older version been used in this test. Asterisk can be downloaded at [40].

6.2.3 Other equipment

Besides the above described hardware and software is a laptop needed. Performance requirements for the laptop are quite low and almost any laptop available on the market today will work. For this testing has an old laptop been used to show that the requirements are low. The used laptop is a Fujitsu Siemens Amilio SI1520 bought in the year 2005. It has an Intel Core Duo 32-bit T2300 processor with a clock frequency of 1.66GHz, 1GB RAM and 120GB hard disk drive. The laptop has been connected to a wireless network in order to get access to the Internet.

To test this attack has three different smartphones been used as victim phones, and SIM-cards from three different Swedish carriers have been tested. The three phones used were: Apple iPhone 4 running iOS 4.3.3, HTC Touch Diamond 2 running Windows Mobile 6.5 and RIM BlackBerry Torch 9800 running BlackBerry OS 6.0. The iPhone was using a SIM-card from the carrier Tele2, the HTC was using a SIM-card from Telia and the BlackBerry was using a SIM-card from Telenor.

In order to debug and analyze network traffic Wireshark has been used. Wireshark is an open source software for packet analyzing and network analysis and is widely used by system administrators for debugging. In this testing has it been used to analyze SIP-traffic and to monitor the GSM packets sent from OpenBTS. However, Wireshark is not required for implementing this attack but it is an excellent tool for troubleshooting when problems occur. Wireshark can be downloaded at [41].

6.3 Modifications

It is quite straightforward to set up the base station, and there are a lot of good tutorials on Internet for doing this, for example was [42] used as a guide for setting up and configuring the system. However, some modifications of the software have been necessary in order to make the attack possible and make it work in a useful manner. First it was required to modify OpenBTS so that it is able to send at the 1800MHz frequency band, since OpenBTS from the beginning was made for sending in the GSM frequency band of 900MHz. It is said that the latest version of OpenBTS does not require this patch anymore, however in this test it was necessary to get it to work. The patch is clearly presented both at OpenBTS homepage [39], and in [42] and is therefore not described any further in this paper.

The second modification was a bit bigger and it relates to registration of mobile phones to OpenBTS and Asterisk. OpenBTS and Asterisk is implemented to identify a mobile phone with the SIM-card's unique id number called IMSI. A mobile phone that tries to connect to the base station sends its IMSI to the base station and the IMSI is used by OpenBTS to identify the phone. This part works pretty much by itself in OpenBTS, but the problem occurs when Asterisk also needs to know the IMSI in order to register the phone at the phone switch. In the original way it is said that the user should look up the IMSI in the logs made by OpenBTS and manually add it to the Asterisk configuration [39]. This is very unpractical for the attacker, who might want to setup the rogue BTS and then leave to let the base station work by itself. Therefore, has OpenBTS been modified to automatically update the configuration files for Asterisk when a new mobile phone is registered to OpenBTS. It is not many lines of code, but it was not trivial to find the right places to insert them. Appendix C: OpenBTS modification contains the source code for the modification made to fix this.

The configuration of OpenBTS and Asterisk for this attack is by itself trivial to figure out after following the available tutorials and is therefore left out from this paper.

6.4 Test result

The testing of the attack using a rogue base station has led to some scary results. Without any bigger problems has this theoretical attack been implemented and it is working in almost a perfect way. All this could be done in only a few days with hardware for less than \$1500. In the following sections are the results stated and described in detail to present the severity of this threat.

6.4.1 Connection

All three test phones did connect to the rogue base station without any problem and without leaving much information to be notified by the victim. By configuring the BTS to accept any carrier were all phones able to connect at the same time, but in this case was the victim given a hint that something might be wrong, since the carrier logo might not be the corresponding for the victim's carrier. This logo can be set to anything in OpenBTS, but if the system allows different carriers to connect will at least some of them get the wrong logo (or no logo at all). This might go undetected by many victims. However, if a certain victim is addressed it can easily be specified so it matches, since the carrier can be read out from the first five digits of the IMSI.

6.4.2 Calls

This testing has showed that internal calls can be made between phones connected to the rogue BTS as well as calls from a connected phone to an external SIP-client and SIP-gateway. Hence, from the victim's point of view everything does work as usual when making calls. Asterisk has also been configured to record all established phone calls into a wav-file. It is recording both directions so the entire conversation is recorded. This recording can also be played while recording, so almost real time eavesdropping is possible as well. This did work with all three test phones without any problem. It has also been tested and verified that by setting a caller id in Asterisk the attacker can call the victim with a spoofed phone number, so the attacker can pretend to be anyone he want to be.

6 Rogue GSM base station

One shortcoming of this system is that the victim's phone is only known in the rogue GSM network, since the BTS never lets the outer world know about the phone. Therefore, it is not possible to make calls from the outer world to the victim. So, if the victim is connected for a long time, he might be suspicious since he does not receive any calls or SMS.

6.4.3 SMS

All SMS that a connected phone sends is intercepted in Smqueue and logged in plaintext to a file or to a console on the server and can hence be read by the attacker. The SMS is then transformed into a SIP-instant message and sent out from Asterisk. It has not been verified in this testing that a SIP-gateway can retransform it to a SMS and send it to the intended recipient, so it is not sure that the SMS would arrive, as it should in a real GSM network.

It has been tested and verified that it is possible to send SMS from the base station to all connected phones and that the attacker can spoof the sender's number to be any number he want. So, an attacker can intercept a SMS sent from the victim, read it and then answer it pretending to be the intended recipient. This did work with all test phones as well.

6.4.4 Over-the-air-programming

In theory would it be possible to create OTAP messages and send these from the base station to all connected phones. The structure of OTAP messages is depending on both the phone model and the carrier and it is not common that the carriers reveal this information to the public. So, even after repeated tries and testing with minor achievements it has not been possible to verify that this work in practice.

6.5 Conclusions

In only a few days the attack equipment was assembled and configured so that this attack can be performed. The knowledge needed to implement this attack is available on many different places on the Internet, so the knowledge requirements of the attacker are actually minimal. This, together with the fact that the equipment is quite cheap nowadays, makes this attack indeed likely to happen in real life. The fact that the needed equipment is small enough to be placed in a backpack makes it easy to get close enough to a specific target, even though the used equipment has a limited range of about 30-40 meters. The distance the BTS can be placed from the victim heavily depends on the signal power of the real base stations at the place, since mobile phones connect to the BTS with strongest signal power. However, if the attacker wants to increase the range, he can buy better antennas and/or a signal amplifier and then be able to cover much larger areas. In Figure 13 below is the attack equipment for this testing shown.



Figure 13: The attack equipment: BTS and laptop

Another thing the attacker might have to deal with is that most phones use 3G whenever it is possible and this attack is not possible against a 3G network. However, 3G signals are transmitted in a different frequency than GSM is, and it is therefore possible to block 3G signals with a signal jammer and in this way make the victim's phone fall back to GSM. 3G jammers can be bought online for only a few \$100, for example from [43].

With only a low level of knowledge and a quite small amount of money can an attacker eavesdrop and record all phone calls a victim makes, read all SMS the victim sends and send SMS and call the victim with a spoofed number. All this can be done without the knowledge of the victim and pretty much regardless about which phone and carrier the victim uses. This threat is really severe and mobile phone users need to understand this. For example, think about the scenario of that the attacker places a rogue BTS at the entrance of a company. This could lead to that all calls made with mobile phones from the company's employees can be eavesdropped on, which probably means that company confidential information is disclosed.

7 Risk analysis

Both iOS and BlackBerry OS have been described and discussed in previous chapters and this part of this paper is about the risk and threat analysis for these systems. This chapter describes the work method and process of the analysis and the result in sense of identified risks and threats in these systems. It is also discussed what vulnerabilities that are identified to cause these threats. This chapter starts by describing and defining some fundamentals of risk and threat analysis. It is important to understand these fundamentals in order to be able to follow this analysis.

7.1 Analysis fundamentals

To be able to follow this risk and threat analysis it is important to understand some common terms and definitions. The most important terms are therefore described below. It is worth to know that some of these terms are defined in a specific way just for this analysis and they are not always used in the most common way. So, even if you as a reader feel quite confident in this field, it might still be a good idea to read these definitions.

7.1.1 Assets

An asset is a system resource that should be protected from disclosure. Assets can be data, services and/or hardware that in some way are important for the system and/or the usage of the system. The device itself is an example of an asset in a smartphone system, since the device is quite expensive and must be protected from thieves [44]. In this analysis is each asset categorized in one of the following categories: hardware, data, firmware, service, communication or financial assets. Some assets might be hard to categorize in just one category, but if this is the case, the most appropriate category is chosen.

7.1.2 Agents

An agent is a person that tries to use or misuse a system in some way so that assets are disclosed. This can be done by purpose or by mistake. There are a lot of different types of agents that interact with systems in different ways and therefore lead to different risks. Examples of agents are careless users, crackers, industrial spies and/or disgruntled employees. It is important to think about all possible agents for a system to be sure to detect all systems risks. Different agents will probably disclose different types of assets, e.g. careless users might lose their smartphone while industrial spies try to get hold of company information [45]. Agents are in many cases also known as attackers and in this paper will the words be used interchangeable. However, the word attacker is often associated with deliberate actions and can make persons forget about the agents that do it by mistake. Therefore, both words are used in this paper.

7.1.3 Threats

A threat is a person, a thing or an action that might harm the safety of certain assets in a system. Threats can be whatever that can threaten a system and its assets, e.g. a thief trying to steal the device, a natural disaster shutting down a service, and so on. It is important to consider all threats of a system to achieve a correct and reliable analysis [44]. In smartphone systems are many threats connected to the fact of disclosure of sensitive information in some sense.

7.1.4 Vulnerabilities

Vulnerabilities are weaknesses in a system that can be exploited by agents in order to disclose an asset and harm the system. Vulnerabilities can among other things be implementation bugs, flaws in system design, flaws in architecture and/or insufficient user knowledge/awareness. Different types of vulnerabilities enable different threats. One of the most well-known and common vulnerabilities is implementation bugs in the

7 Risk analysis

form of buffer overflows. So, vulnerability is a weakness that can be exploited by an agent to attack a system in order to get hold of some system asset [44].

7.1.5 Complexity

Complexity is a value that is assigned to each possible attack and it is used for indicating how complex it is to perform the attack. (The complexity value is later used for calculating a risk factor). The complexity value is depending on many things and it is therefore not trivial to assign complexity values to attacks. Things to consider when determining the complexity is how much time it takes to establish the attack, how much knowledge is needed to perform the attack, how much the needed attack equipment will cost, what the probability that the attacker will get caught is, and so on. The scale for the complexity value is depending on how the risk factor calculation is performed. Both the scale and the meaning of the values can differ between different analyses. In this analysis will a low complexity value represent that it is hard to perform the attack and a high value represent that it is easy. This will be discussed in greater detail in section 7.2 below.

Let's give an example to clarify how to reason, so let's address a smartphone attack in the sense of stealing the device. There are of course different ways to steal a device, e.g. pickpocketing, break in or robbery. Let's look at pickpocketing. Pickpocketing does not cost that much for an attacker, since no equipment is needed. It is neither that hard to follow a specific victim. The time it takes to perform the attack is not that long, but probably the agent must practice for a while before he is able to do it. However, there is a quite high probability that the agent will get caught and the consequence of getting caught is quite high. So, it is quite easy to perform the attack, it does not cost much to do it, but it needs some knowledge and there is a high risk of being caught. All this must be considered when determining the complexity value.

7.1.6 Impact

The impact is the resulting consequence of a successful attack on a system. The impact can also be said to be the damage on the system that a threat makes when it happens. Impacts can be of different types, e.g. financial loss, confidentiality breach, integrity violation and damage of reputation and so on. Each impact is usually assigned an impact value that indicates how severe the impact is for the victim. The impact value must be considered separately for each type of victim since the severity of an attack often differs for different types of victims [45]. For instance is loss of a smartphone device, in the sense of financial value, not as severe for a large company as it is for a regular user. On the other hand it is probably much more severe for a large company to lose a smartphone in the sense of losing sensitive information than it is for a regular user. The impact value is used to calculate the risk factor as well.

7.1.7 Risks

A risk is the combination of a threat and an attack. It represents the risk of that a threat is carried out by a specific attack. The risk factor is a combination of the complexity of an attack and the impact of the attack when it is successful. It is a benchmark value showing what is possible and probable to happen to a system. There are different ways to calculate risk factors using the complexity value and the impact value, but in this analysis the factor is calculated by multiplying these values. (The calculation for this analysis is described in greater detail in section 7.2 below).

The risk factor is showing how severe and how probable an attack is, and therefore this factor can be seen as a priority for security personnel. The risk with highest risk factor should be resolved first in order to make the system most secure in shortest time [45].

7.2 Method and work process

This risk analysis for iOS and BlackBerry OS has been performed in seven steps for each of these systems. iOS and BlackBerry OS has been analyzed separately and for each system has two different use cases been addressed in order to make the analysis as reliable and correct as possible. This has been done since different

types of users handle their device in different ways and it leads to that different assets might exist and it might also affect the complexity and impact of an attack. The two use cases that have been considered are business usage and regular usage. Regular usage means that the user is an individual that uses the device for personal use only and the devices will therefore not contain any corporate information. The other use case, business usage, means that the device is used for business usage and is tightly connected to corporate information. It is today very common for business users to also use their devices for personal use, but in that case the device is considered to be a business device in this analysis.

As mentioned above the work process of this risk analysis is divided into several steps for each system and these steps are:

1. Asset identification
2. Threat identification
3. Attack identification
4. Complexity value determination
5. Impact value determination
6. Risk identification and risk factor calculation
7. Vulnerability identification

The first step is asset identification and it is about determining which assets that exist in the analyzed system. It is important to consider every asset there is to achieve a reliable risk analysis. In this analysis all assets will be categorized in a category to ease the risk analysis. The second step is about determining what threats that exist for each asset in the system. The third step is about considering all possible attacks for each of the found threats. This step and the following steps are really big steps and much of the time of the risk analysis has been spent in these steps.

Step four and five is about determining the complexity value for the possible attacks that was identified in the previous step. The complexity value in this analysis is an integer value in the range 1-5, where 1 represent the highest complexity and 5 represents the lowest complexity. The impact value is also represented by an integer value in the range 1-5, where 1 represents the smallest impact and 5 represent the highest impact. Each value of these terms is associated with a level, which describes them. The ranges and their corresponding levels are shown in Table 5 below. As stated above the impact and complexity will be discussed for both business usage and regular usage. This is done since these values might be quite different for different use cases.

Table 5: Complexity and impact levels in this analysis

Value	Complexity level	Impact level
1	Very high	Barley notable
2	High	Minor
3	Medium	Significant
4	Low	Severe
5	Very low	Catastrophic

Step six is about considering the previously found threats and attacks that make up risks for the system. It is also about calculating risk factors for all risks. In this analysis the calculation is done by multiplying the complexity value with the impact value and the result is the risk factor. The risk factor is assigned a level to describe it instead of just using the integer value. In Table 6 below are levels for the possible risk factors shown. The last step is about extending the risk analysis to recognize what vulnerabilities that make each attack possible. When all these steps are performed it should have resulted in identified risks, calculated risk

7 Risk analysis

factors and identified vulnerabilities for each risk found in the system. This is done separately for iOS and BlackBerry OS, but in the following sections the results are discussed for them together in the case of equal results; for example both systems have almost the same assets and it is therefore presented as a common result instead of repeating it in this paper.

Table 6: Risk levels in this analysis

Risk factor	Risk level
1-4	Negligible risk
5-8	Low risk
9-12	Intermediate risk
13-19	Important risk
20-25	Critical risk

7.3 Asset Identification

There are a lot of different assets in smartphone systems, which should be protected from disclosure. Most assets are data assets of different kinds but also hardware, service, financial, communication and firmware assets have been identified. Most of the assets are obvious but a few of them might need to be explained in order to understand why they should be considered as assets. The following identified assets are common for both iOS and BlackBerry OS devices.

7.3.1 Data assets

A smartphone is packed with data that is more or less sensitive for the user. Assets such as stored passwords, emails, messages, stored encryption keys, contacts and payment credentials exist on most smartphones today. These assets are of course sensitive for the user and should be protected. In most smartphones there also are calendars, notes and images that should be considered as important assets. For example the calendar events are important information for an agent that wants to surveil a user or an industrial spy that is spying on a company. Images are probably a more sensitive asset for private users than for business users since regular users probably use this feature more than business users do. The more non-obvious data assets identified in smartphones are stored GPS-coordinates for visited locations, the call history and the keyboard cache. The first two assets are of great use for an agent that tries to surveil a user. The keyboard cache is not that well known by users, but it exists in many of the current smartphones. It is a cache of almost all words ever typed on the keyboard and it is saved in order to be used for auto-correction of commonly typed words. Passwords are not saved in this cache file, but it can contain a lot of sensitive information and this file should not be leaked to an agent.

7.3.2 Hardware assets

There are a few hardware assets in smartphone systems as well. The smartphone itself must be considered as an asset since it is both expensive and contains a lot of sensitive data that must be protected. The microphone on the smartphone is also an important asset since it can be used to record and eavesdrop on both conversations in phone calls and on conversations between people close to the device. Almost all smartphones on the market today have a built in GPS-module that can establish the device location at all times. This hardware must be considered as an asset to protect, since it will ease user surveillance if an agent gets the device's location at all time. The last hardware asset identified in smartphone systems is the baseband module. Since the baseband handles all signal management for a device, it is an important asset. An agent getting access to the baseband could eavesdrop on conversations, make services unavailable and so on.

7.3.3 Firmware assets

There is not that many identified firmware assets in these systems, but they are very important assets. The first identified asset is the bootloader. The bootloader's task is to load, initiate and start the operating system on the device and also to check that the operating system is a correct, untampered and valid operating system. So, if an agent can modify the bootloader, it could be changed to load a modified operating system containing severe security flaws and backdoors. Therefore, the bootloader must be considered as an important asset that must be protected from tampering. The other identified firmware asset is drivers. The kernel of the operating system contains a bunch of drivers that are needed for the device to work as it should. Many of these drivers are running with full system rights so that they can work as they should. So, if a driver can be modified to contain malicious code then this code will be executed in kernel mode with full root access and it can do almost anything with the system. Therefore, drivers must be protected and checked so that they are valid before execution.

7.3.4 Communication assets

There are some communication assets in smartphone systems as well. Identified assets are voice calls, SMS/MMS messages, Wi-Fi data and Bluetooth data. Voice calls and messages should be considered as assets to protect, since personal and/or business phone calls and messages can contain sensitive and confidential information that not should be leaked to any unauthorized persons. Even Wi-Fi and Bluetooth data transfers to and from a device should be considered as an asset since it can contain sensitive data.

7.3.5 Financial assets

There is only one financial asset for smartphone systems identified in this analysis and it is the phone bill, since a smartphone is tightly connected with the user's financials in the sense of calls and services billed to the users invoice. So, if an agent can perform expensive calls from the device it can be very expensive for the user. It is quite common today that other services can be paid by charging the phone bill as well, for instance traveling tickets often are possible to pay for via SMS and so on.

7.3.6 Service assets

The last category of assets included in this analysis is service assets, and they are perhaps the most abstract assets to think about. The two identified assets in this category are availability and VPN-connections. As described earlier in this paper is availability about that the device and its services, i.e. phone calls, messaging, web browsing and so on, is available and working when the user needs to use them. The VPN-connection should be considered as an important asset since it is by definition a remote way into an internal network. So, if this service can be accessed by an agent, it will threaten the security of the entire internal network.

7.4 Threat identification

There are a lot of threats in smartphone systems that are threatening the safety of system assets. In this analysis six threats have been identified and considered. In the following sections are each of these threats explained and motivated. All identified threats are valid for both iOS and BlackBerry OS and the impact if a threat happens is the same for both systems, but it is depending on the use case. The impact value is determined for both use cases and motivated for each threat as well.

7.4.1 Data leakage

The first identified threat is the threat that sensitive and/or private data stored on the device is leaked to an agent. This includes data such as emails, contact information, calendar events and so on. (Please note that in this paper user credentials such as credit card numbers and passwords are not included in this threat, it is

7 Risk analysis

instead discussed in section 7.4.5 below). This is a severe threat since a smartphone contains on the average a large amount of sensitive data that should not end up in the hands of unauthorized persons. This threat can of course be carried out in many different ways, but it is regardless of the attack type a severe threat. The impact level is determined to be valued as severe for regular usage, since a smartphone contains a large amount of sensitive data. For business usage has the impact level been valued as catastrophic since business devices usually contain even more sensitive information than a regular user device.

7.4.2 Financial loss

This is the threat of financial loss in any way for the user. This includes losing the expensive device itself, getting billed for expensive calls and messages or being signed up for pay services. This is of course a threat since the financial value that the user loses can be big and it will hurt the user in many ways. The impact level is valued as catastrophic for regular users and significant for business users. The impact level is considered higher for regular users, since it in many cases is more severe for a single person to lose a big amount of money compared to the same amount of money for a business.

7.4.3 User surveillance

The threat of that a user is being surveilled by an agent can be carried out in a lot of different ways in smartphone systems. It includes ways such as eavesdropping on phone calls, reading emails, locating users with GPS-coordinates and so on. All these are threats for the user's privacy and integrity as well as it might reveal sensitive corporate information. The impact level is valued as severe for regular users since their privacy and integrity are violated and the information about the user location can be very sensitive, for instance can such information be used for burglars in order to know when to break into the victim's home. For business users the impact level is considered to be catastrophic, since besides the user's and company privacy, confidential corporate information's integrity can be violated as well.

7.4.4 Device and/or service unavailable

Smartphones are used by their users for many different services, such as to make calls, send/read emails, browse the internet and connect to corporate networks and so on. In many cases the user expect does that it work when he needs it. Therefore, there is a threat that an attacker makes important services unavailable, so the victim cannot perform the action he requires at the moment. There are lots of ways to make a device and/or a service unavailable, and this is therefore a threat that must be considered. Impact level for regular users is set to minor, since in most cases would it just be annoying for the victim that he cannot use the service right then, but it is not that severe. For business users it can be more important to be able to use a service at a specific time, since missed calls and emails for instance can lead to loss of income as well as damage to reputation for the business. Therefore, the impact level is valued as significant for business usage.

7.4.5 Credentials leakage

This is the threat that a user's credentials are leaked from the device to an attacker. Smartphones often contain credit card credentials as well as authentication credentials for web sites, email servers and other computer systems. Besides the possibility that these credentials are stored on the device there is also a risk of leakage of them, when the user enters them while browsing the Internet. This information is indeed sensitive and should be kept safe. If the attacker gets his hands on this kind of information entire corporate networks and servers can be attacked as well as there being a risk for huge financial loss for an individual. Therefore, the impact level for this threat is considered to be catastrophic for both regular users and business users.

7.5 Attack identification

There are different ways to attack smartphone systems depending on what the attacker wants to achieve. As described in section 2.3 there are four main categories of attacks that are focused on in this thesis work, i.e.

remote access, physical access, malware and social engineering attacks. This section is focusing on attacks on iOS and BlackBerry OS devices and is discussing some important attack types for each of these two systems. For each of the following attacks both iOS and BlackBerry OS are evaluated to see if the specific attack is possible and how complex the attack is in each system.

7.5.1 Jailbreaking/Rooting

As described earlier in this paper jailbreaking is the process of gaining root access for an iOS device. This gives the user full access to all features and capabilities of the device, and it bypasses almost all the security measures implemented in the system, e.g. that only signed applications can be installed, restricted access to baseband module and so on, is bypassed. The word jailbreaking is associated with iOS devices so for BlackBerrys the word rooting is used instead. The meaning is however equivalent. To be able to do jailbreaking/rooting is a known vulnerability exploited with specific software tools. The whole process usually only takes a few minutes to do and it does not matter if the device is locked or not[46].

Gaining full root access to a device is highly valuable for an attacker and therefore is the possibility of jailbreaking/rooting a device a severe attack for the system security. If an attacker can get physical access of a device, he only needs a few minutes to be able to do almost anything with the device. For the current version of BlackBerry OS there is no known way to root devices, and therefore is BlackBerrys so far spared from this attack in the current version. For the current iOS is a jailbreak released on the Internet, accessible for anyone who wants to do it, and therefore is jailbreaking a scary attack for iOS devices.

The complexity level for this attack is considered to be low for both regular users and business users with iOS devices. There is no known possible way to root a BlackBerry OS device at the moment, but it is not proven that it is impossible and therefore is the complexity level evaluated as very high for BlackBerry OS devices regardless of use case.

7.5.2 Malware attacks

Both iOS and BlackBerry OS have some implementations for preventing installation of malware and to minimize what a malicious application can do. Despite this has both systems suffered from malicious applications that upload sensitive information from the device. iOS is relying on that only signed applications can be installed and that all signed applications are reviewed by Apple [11]. The huge amount of applications and the fact that no review of the source code is done still makes a few bad apples pass this review. A good thing with Apple's way of distribution is that it makes it almost impossible for a malware to spread to other devices by itself. That is because even though a malware by mistake has been signed by Apple it must still be downloaded from App Store and cannot be passed from device to device.

BlackBerry OS does not require applications to be signed if they do not use protected API:s and therefore can applications more easily be distributed for BlackBerrys than for iOS devices [22]. It is possible to access much sensible information using only public API:s, and therefore is malicious applications a real threat for BlackBerrys. Such malicious applications can also spread by emails or by SMS that include links to a download site for the application.

As shown in chapter 5 it is not even that hard to implement a malware for these systems using only public API:s. In fact all the needed information about how to do it is quite easy to find on the Internet. Therefore, malware attacks must be considered important and possible for both iOS and BlackBerry OS devices.

The complexity level is considered to be high for business users and medium for regular users with iOS. This is because business users usually do not install as much strange software on their device as the regular users do. For BlackBerry OS is the complexity level valued medium for business users and low for regular users. It is considered easier to attack a BlackBerry device than an iOS device, since applications can be distributed and installed more freely in BlackBerry OS.

7.5.3 Phishing attacks

Phishing is a type of social engineering attack that is about tricking a user to reveal sensitive information over the Internet. With phishing attacks the attacker is pretending to be someone or something else in order to trick the user. An email where the attacker pretends to be the victim's bank asking for the victim's account information is an example of a phishing attack [47]. Smartphones are often used for browsing the Internet and therefore is phishing sites a possible way to attack these systems; and this includes both iOS and BlackBerry OS. When a user browses the Internet on a regular computer there are warning signs to look for in order to detect phishing sites, e.g. sites using SSL has addresses starting with HTTPS and a small pad lock icon is shown at the bottom of the browser. In neither iOS nor BlackBerry OS are these symbols always visible because of the small display on the device. This leads to that phishing sites are much harder to detect on these devices and the user can more easily be tricked to disclose sensitive information [47].

The complexity level is determined to be low for both use cases and both systems since neither of these systems provides any specific protection for phishing attacks and both systems are lacking in the same way for example by not showing SSL-symbols and warnings.

7.5.4 Attacks on encryption

Both iOS and BlackBerry OS are relying on encryption for much of their security and for protecting sensitive data on their devices. Both systems are using standardized and secure encryption algorithms, which seem to be implemented in a correct and reliable way. So, the cryptographic of these systems should be secure and make it really hard for an attacker to access data on devices. However, an important thing to remember is that encryption key management is a critical point of cryptology. So, even if a highly secure algorithm is used, it will not provide much security for the system if the key is public available or poorly generated. In both iOS and BlackBerry OS are most encryption keys generated on the device without any user interaction, and therefore is the security of their cryptology heavily depending on how well these systems are generating keys.

Apple is following their policy of not commenting or publicly documenting how their system is working, and it is therefore extremely difficult to know exactly how their encryption keys are generated. What is known is that the device encryption key is generated only from data stored on the device and the data protection key is generated from the device passcode/password in some way [8]. It is also stated that the data protection key is not stored on the device at all, which leads to the conclusion that the generation does not depend on any timestamp or something that might change over time. Therefore, it is probably only using the passcode/password as input for the generation. By default iOS is only using a four-digit passcode when activating data protection, and generating encryption keys based only on a four digit input is not sufficient, since exhaustive testing of the possible combinations of four digits is done really fast. It has also been shown that with physical access to an iOS device it is possible in a short time to jailbreak the device and execute a script on the device which, tells the device to transfer all data on the device to a computer. Since the device has access to the device encryption key it will transfer the data in plaintext to the computer and in this way the encryption can be bypassed in iOS [9].

The generation and storage of encryption keys in BlackBerry OS is previously described in section 4.3 in greater detail. Most of the keys are generated by using hashed inputs to the DSA pseudorandom number generator. The input is in most cases also depending on the current time when the encryption key is generated. This way of generating encryption keys is consider as being quite secure, since the generated strings have a high level of entropy and it will be almost impossible to successfully perform exhaustive testing [48]. It is however important to keep the secret used for the seed securely stored. In BlackBerry OS the encryption keys and the key generation secrets are stored encrypted by using the ephemeral key. So, if the ephemeral key is compromised, the entire cryptology security is compromised as well. The ephemeral key is generated from the device password using the PKCS#5 algorithm [22]. If the password is secure enough, this is sufficient enough for the security but once again the entire security is depending on secure passwords.

Note that this attack is considered to be performed by an attacker that has physical access to the device. For

iOS devices the complexity level of encryption attacks is considered to be high for business users and medium for regular users. The difference here is depending on that business users in many cases use better passwords than the regular user. The complexity level for BlackBerry OS is considered to be very high for both use cases.

7.5.5 Attacks on GSM/3G

Smartphones are usually operating in the 3G network whenever they are in reach of a 3G network, but most smartphones are still able to use the older network GSM, which still has better network coverage in most countries. Most smartphones are configured to switch to GSM automatically when no 3G connection is available and to automatically connect to the base station that provides the strongest signal power. iOS and BlackBerry OS devices have this default behavior, so they are switching to GSM when it is needed. This is of course good for the usability and availability of the device and therefore this behavior is still encouraged. The problem is that 3G is a much more secure network than GSM, so devices switch to a less secure network without notifying the user.

The German security researcher Karsten Nohl has shown that GSM is deprecated and not secure to use since the GSM encryption can be cracked in a reasonable time using only equipment that is widely available [49]. The needed equipment is not even that expensive anymore. Another issue with GSM is that it supports communication without encryption if the base station requires that no encryption should be used. The real problem with this is that almost no mobile device on the market is notifying the user when GSM communication is sent unencrypted. Neither iOS nor BlackBerry OS devices notify the user about this. So, an even easier way to attack the GSM network is to set up a false base station, place it close to the victim so it provides the strongest signal power and then let this base station tell the devices to not use encryption. It will result in that all GSM traffic is passing through the attacker's base station unencrypted. The required equipment for this is available to buy on the public market and the needed source code is available on the Internet as open source projects [39]. This kind of attack using a rogue base station has been tested in this thesis work and chapter 6 describes the attack and the result in greater details.

Since devices always try to use 3G as a first choice, an attacker must disable 3G to be able to attack devices through GSM attacks. This is easily done by buying a 3G-signal blocker, which also is publicly available on the Internet for only a few \$100 [43]. As seen in the rogue base station test attack performed in this thesis both iOS and BlackBerry devices are vulnerable to this kind of attacks. Therefore, GSM/3G attacks are eminently important to consider in both iOS and BlackBerry OS.

The complexity level for GSM attacks is evaluated to be medium for both systems and use cases, since the attack with a rogue base station is both effective and quite easy to perform today. Other attacks on GSM are a bit more complex, but it is still not hard to do.

7.5.6 Hardware tampering

Earlier in this chapter it was mentioned that the attacker can use jailbreaking/rooting as an attack, if the attacker can get physical access to the device, and this can lead to disclosure of all data on the device. Besides that an agent can attack the system in other ways if physical access to the device is granted him. For instance the hardware of the device itself can be tampered with by an attacker. For example the battery can be exchanged for another battery, which looks normal but contains a microphone used by the attacker to eavesdrop on the victim. Another possibility is that the entire device is exchanged for another one that looks the same and so on. Even though jailbreaking/rooting does not take that much time this attack can be even faster and needs even less time for an attacker to perform it. iPhone is probably a little safer against hardware tampering than BlackBerry devices are, since it requires a screwdriver to be able to remove or change any hardware (except for the SIM-card) while many BlackBerry devices can change the battery by removing the back of the device without any tools [24]. This is however only requiring a small amount of extra time for iPhones, so it is still totally possible to do (and it might even make the attack harder to detect)[6].

Hardware tampering is easy to perform and does not take long time to do. As stated above it is a bit harder to

modify the iPhone hardware than BlackBerry's and therefore is the complexity level for iOS for both use cases determined to be low and for BlackBerry are both use cases determined to be very low.

7.5.7 Spoofed Wi-Fi access points

Many smartphone users connect their devices to available Wi-Fi access points to get a faster internet access than they get when using any mobile phone network. There are a lot of available Wi-Fi networks in public areas today, e.g. at train stations, airports, shopping centers and so on. Some of them do not use any encryption at all and data is transferred in plaintext in such networks. For an attacker it is not hard to set up an access point at a public place and let unaware victims connect to it, which gives the attacker a man-in-the-middle position. This attack is of course depending on that the victim has low awareness about the security of Wi-Fi networks, so they connect to an unknown network, but there are ways for an attacker to make it less obvious for the victim. Since the network can be named anything the attacker can name his rogue network the same as another available network is called, and in this way act as the original network. This attack lets the attacker be able to eavesdrop on data traffic from the victim's device and can use techniques like SSL-downgrade to be able to collect sensitive user information [50]. Even spoofing DNS-queries is possible since the attacker can let his access point answer DNS-queries with false addresses, which makes phishing attacks easier to perform.

This attack has been tested in this thesis work. It was tested on a bus trip from Linköping to Stockholm (about 3h traveling) where a laptop was configured as a Wi-Fi access point using a 3G-modem. No sniffing of data was performed, it was only a check of how many devices that connected to the access point, (in fact all devices were disconnected immediately after establishing connection). Not less than seventeen devices connected to the access point and nine of them had device names that indicated that the device was a smartphone. However, only three computers were visible on the bus so probably as much as fourteen devices were smartphones. According to the bus driver there were eighty-six persons on this travel so almost 20% of the persons on the bus did connect to an unknown insecure Wi-Fi network.

Both iOS and BlackBerry devices let their users connect to any Wi-Fi network available, so this attack is possible on both systems. In fact both systems save networks to a list of trusted networks. The complexity for this attack is also equal for both systems, but is considered to be different for the two use cases. For regular users the complexity level is considered to be very low since most regular users do not care about that they use an unknown network. Business users might be a bit more cautious so it is probably a bit harder to get them to use the spoofed network, therefore the complexity level is considered to be low instead of very low.

7.5.8 Network attacks

Smartphones are usually pretty much connected to internet via 3G or Wi-Fi twenty-four hours a day. This makes them possible targets for attacks over internet. Root access to the device is hard to get over internet on these devices, but it has been shown that it is possible. Both iOS and BlackBerry OS use the web-browser engine WebKit which has contained a lot of implementation flaws in the past and new flaws are found every month. Some of these flaws have been proven to be possible to exploit for an attacker using a malicious crafted web site [51] which means that if a victim is browsing a malicious web site the attacker will be able to execute code on the device with root access.

Smartphones work like regular computers and they must have network ports open to be able to communicate. These ports can be scanned with a port scanning tool, e.g. Nmap [52], just as with a computer in order to look for open ways into the system. Port scanning of an iPhone 4 has been done in this thesis work and the result was that only one port was open and it was port 62078, which is said to be used for the possibility of remotely wiping the device. The number of open ports on a BlackBerry depends on the model and what applications that the device is currently holding, since some (signed) applications are able to open ports on BlackBerry devices [22].

It is possible to remotely attack a device over internet, and this has been done for both iOS and BlackBerry OS devices, but it is still hard to do it. It requires high knowledge about these systems from the attacker and a

complex exploit must be written that bypasses many security features in order to have a successful attack. The knowledge requirement together with that RIM and Apple are quite quick to fix the vulnerability prevent attackers with less knowledge from being able to exploit the vulnerability. This leads to that it is quite complex, and quite few attackers can perform this kind of attack. Therefore, is the complexity level valued as high for both systems and both use cases.

7.5.9 Theft

Smartphones are small and mobile by definition and most models are quite expensive. These properties make them valuable targets for thieves and their size also makes them easy to misplace. Besides the threat of that thieves can steal them for the financial value, it is also possible that a thief steals them in order to get his hands on the information that the device might contain. Especially iOS devices are potential targets for this, since it has been proven to be easy to retrieve the information by jailbreaking the device, as described in section 7.5.4 above. Possible ways to steal a device is for example burglary, robbery or pickpocketing.

iOS devices and BlackBerry OS devices are small enough to be easy to steal and can easily be carried away by a thief, but theft is considered as a serious crime and the punishment for an attacker if he gets caught is quite significant. Business users are often more cautious and protective of their devices and therefore is the impact level considered to be little higher for them compared with regular users. The complexity level is therefore determined as medium for business users and low for regular users regardless of the system.

7.6 Risk identification

Above have a number of possible attacks on both iOS and BlackBerry OS been identified. Actually are all attacks possible in both systems, even though the complexity differs between them. Some threats have also been identified and the impact if the threat actually happens has been evaluated. These threats can occur in different ways but still lead to the same impact. Therefore, the identified threats are here merged with the possible attacks and the results are risks for these systems. The risk factor is calculated by multiplying the complexity value with the impact value. The risk factor is a benchmark value of how severe and probable the risk must be considered. A low complexity value together with a high impact value gives the highest risk factor. This means that an attack that is easy to perform and leads to much damage is the most probable thing to happen and is therefore most severe. Not less than 19 risks have been identified in this analysis and each of these risks is described and motivated below. Risk tables are also available in Appendix D: Risk tables, and these give an easy way to get an overview of the identified risks together with which vulnerability that enables the risk. It is strongly recommended for the reader to have a look at these tables before continuing reading this section. For easy referencing between the description below and the tables is each risk numbered, but notice that this integer value only is a reference number, and it has nothing to do with the severity or probability of the risk.

Risk 1: It is the risk that sensitive and/or private data is leaked to an attacker that has physical access to the device and jailbreaks/roots it. This is of course a severe risk for iOS devices because they are easy to jailbreak and it is proven that a huge amount of the stored data can be extracted in this way [9]. The risk level for iOS devices used for business usage is calculated to critical risk and the risk level for regular usage is calculated to important risk. The vulnerabilities that enable this risk are the existence of implementation bugs that can be exploited to gain root access together with the fact that a careless user lets his device out of sight so an attacker can get physical access to the device. There is no publicly known way to root a BlackBerry OS 6 device and hence this risk is not that severe for this platform. The risk level is therefore calculated to low for business usage and negligible for regular usage.

Risk 2: It is the risk that sensitive and/or private data is leaked to an attacker through a malware that is installed on the device. As shown with the proof-of-concept spyware made in this thesis work a malware can access a lot of sensitive data and upload this data to a server, see chapter 5 for more details. Even though the spyware was made for iPhone, it is equally possible to do for a BlackBerry and it is even easier to distribute applications (and malwares) for BlackBerry OS. The risk level for iOS is calculated to intermediate risk

7 Risk analysis

regardless of the use case, and the risk level for BlackBerry OS is calculated to important risk for both use cases. The vulnerabilities that enable this risk are the insufficient malware protection in both systems and also the unawareness by the user about these threats. Many users still do not know that smartphone malwares exist and what they can do, and this leads to the fact that users are not as cautious as they should be when installing applications.

Risk 3: It is the risk that sensitive and/or private data is leaked to an attacker through phishing attacks. Phishing attacks on smartphone systems include spoofed web sites, fake emails and so on, just as with other computer systems. Phishing attacks are not that hard to perform, and that makes this risk important to consider. It is probably hard for an attacker to trick the user to reveal a large amount of data, but the sensitivity of the data can be high anyway. The thing that makes smartphones more vulnerable is the fact that the displays on the devices are much smaller than a regular computer screen and can therefore not contain as much information as a regular screen. Therefore, some common security symbols are not present on smartphones, which makes it harder for the user to recognize a fraud. This together with the unawareness of the user for this kind of attacks is the vulnerability that makes these attacks possible and successful. It is equally easy to perform these attacks on iOS and BlackBerry OS. The level of sensitivity of leaked data can be higher for business usage than regular usage, and therefore the risk level is calculated as critical risk for business users and important for regular users regardless of the system.

Risk 4: It is the risk that sensitive and/or private data is leaked to an attacker that has physical access to the device and cracks the device encryption. The device encryption on BlackBerry OS is quite strong as mentioned earlier, and as long as strong passwords are used it is almost impossible to break the encryption in any reasonable time. Therefore, the risk level for business usage with BlackBerry OS is calculated to low and for regular usage it is calculated to negligible. The key generation and management in iOS is not as good as it should be, and the encryption can be bypassed [9]. Therefore, the risk level is calculated to intermediate risk for both use cases with iOS.

Risk 5: It is the risk that an attacker accesses the device remotely over internet and extracts sensitive and/or private data from the device. Both iOS and BlackBerry OS devices are quite well protected against network attacks like these. It is possible, and it has been shown that for example maliciously created web sites can lead to data leakage [51]. However, these attacks are quite complex and require a lot of knowledge from the attacker, and this makes the attack less probable to happen. The risk level is calculated to intermediate risk for business usage and to low for regular usage regardless of the system. The vulnerabilities that enable these attacks are implementation bugs and misconfigurations. Implementation bugs are common in the browser engine WebKit which both iOS and BlackBerry OS use. All though patches for known bugs are released often there are still bugs left to be found that might be exploitable for network attacks.

Risk 6: It is the risk of financial loss for the user by an attacker stealing the device. Smartphones are expensive and valuable and it is therefore a risk of financial loss if the device gets stolen. The risk level is calculated to intermediate risk for business users and critical risk for regular users regardless of the system. That is because the finances of an individual are usually much less than for a company. The vulnerabilities that enables this risk are the fact that the devices are small, mobile and easy to walk away with together with the fact that the device is expensive. Even careless users are a part of this vulnerability since a careful user probably makes it much harder for a thief to get the opportunity to steal the device.

Risk 7: It is the risk of financial loss for the user by getting billed for expensive calls, messages and/or services made by a malware. This is possible to happen if a malware is installed on the device and as discussed in earlier threats malware attacks are possible in both systems. The risk level is calculated to low for business usage and to important for regular usage with iOS. For BlackBerry OS the risk level is calculated to intermediate for business usage and to critical for regular usage. The vulnerabilities are once again insufficient malware protection together with user unawareness.

Risk 8: It is the risk of financial loss for the user by an attacker remotely accessing the device and placing calls and/or sending messages. Gaining full access to a device remotely is as stated before quite hard and remotely placing calls is not trivial either. However, it is possible and the severe impact for regular users makes this a risk that must be considered. The risk level is calculated to low for business usage and to

intermediate for regular usage regardless of the system. It is still implementation bugs and misconfigurations that enable remote attacks.

Risk 9: It is the risk that the user can be surveilled by an attacker using a spyware installed on the device. Spyware can upload information from the device, look up current GPS-coordinates and record conversations with the microphone and much more. These things make it easy for an attacker to surveil and stalk the user. Since it is more difficult to infect iOS than BlackBerry OS the risk level is calculated to intermediate risk for iOS and important risk for BlackBerry OS regardless of the use case. Vulnerabilities are insufficient malware protection and user unawareness.

Risk 10: It is the risk that the user is surveilled by an attacker which eavesdrops on calls and messages from the user's device. As shown in the practical testing part of this thesis work it is possible to attack GSM and eavesdrop on a victim's phone calls, see chapter 6 for details. It is quite easy to perform this attack and even other attacks that include breaking the GSM encryption, and it leads to that all calls made can be eavesdropped on. The risk level is calculated to important for business users in both systems and to intermediate for regular users in both systems. The vulnerabilities that enable these attacks are the fallback to GSM and the fact that GSM contains design flaws. The trust users have for GSM are also a vulnerability, since most users do not know about the flaws in GSM.

Risk 11: It is the risk that the user is surveilled by an attacker that has tampered with the victim's device, e.g. planted a bugging device on it. With physical access to a device it is quite easy to tamper with the device. The hardest part here is to get physical access for the short time needed for the modification and therefore is the biggest vulnerability that enables this risk careless and unsuspecting users. Together with the fact that parts are exchangeable and sometimes also easy to remove is this risk possible. The risk level is considered to critical risk for both use cases with BlackBerry OS and it is considered to critical risk for business users and to important for regular users with iOS.

Risk 12: It is the risk that an attacker makes the device unavailable when needed by the user by using malwares. The risk level is calculated to be low for both use case with iOS and intermediate respectively low for business usage and regular usage with BlackBerry OS. The vulnerabilities that enable this risk are insufficient malware protection and user unawareness.

Risk 13: It is the risk that an attacker makes calls and messages unavailable when needed by the user by using 3G/GSM attacks. There are many ways to stop calls and messages to and from a device by attacking the mobile phone network. For instance can network signals be blocked with a radio jammer [43]. A rogue base station can also be used since the base station can be configured to not connect any calls or messages at all. The risk level is calculated to be intermediate risk for business usage and to be low for regular users regardless of the system. The vulnerabilities that enable this risk are the usage of GSM and the GSM design flaws together with the sensitive network signals that easily can be blocked.

Risk 14: It is the risk that an attacker makes the device or certain device services unavailable when these are needed by tampering with the device. Such tampering can be done by removing the battery, destroying the device, disabling buttons and so on. The vulnerabilities that enable this risk are once again careless users and exchangeable parts. The risk level for iOS devices is calculated to be intermediate risk for business usage and low risk for regular usage. For BlackBerry OS devices the risk factor is calculated to important for business usage and intermediate risk for regular users.

Risk 15: It is the risk that user credentials are leaked to an attacker by malware installed on the device. User credentials are in many cases both stored on the device as well as entered on the device while browsing the internet. Malwares, such as keyloggers, can log this sensitive information and send it to the attacker. The risk level is calculated to intermediate for business usage and to important for regular usage with iOS. For BlackBerry OS it is calculated to important for business usage and to critical risk for regular users. Vulnerabilities are insufficient malware protection and user unawareness.

Risk 16: It is the risk that user credentials are leaked to an attacker by phishing attacks. Collecting user credentials is pretty much exactly what phishing attacks usually do and this risk is certainly important to consider in smartphone systems. The risk level is calculated to critical risk for both use cases and systems.

7 Risk analysis

Vulnerabilities are user unawareness and small device displays.

Risk 17: It is the risk that user credentials are leaked to an attacker by using a spoofed insecure Wi-Fi network. Neither iOS nor BlackBerry OS forbids the use of unprotected Wi-Fi networks, and they do not warn for the risk with this either. Hence, it is up to the user to choose if he wants to connect to the Wi-Fi network or not. So, the vulnerabilities that enable this risk are user unawareness together with inadequate warnings from the device. The risk level is calculated to critical for both use cases with both systems.

Risk 18: It is the risk that user credentials are leaked to an attacker through remote access to the device. The risk level is calculated to intermediate risk for both use cases with both systems. It is not considered as severe because of the high complexity needed for the remote access attack. Vulnerabilities are implementation bugs and misconfigurations.

Risk 19: It is the risk that the user is being surveilled by an attacker who uses a spoofed Wi-Fi access point in order to eavesdrop on data traffic. By spying on the victim's network traffic much information about the victim can be revealed for the attacker. The risk level is calculated to critical for both use cases with both systems. The vulnerabilities are again user unawareness and the possibility to accept insecure Wi-Fi networks.

7.7 Analysis result

This analysis resulted in 19 identified risks for both systems. Some of them are severe risks and others are mild ones. However, it is clear that both systems contain vulnerabilities that make them insecure in some senses and a smartphone user must understand the available risks. One conclusion that can be made from the result of identified risks is that many risks are severe because of the unawareness or carelessness of the user. For instance, many of the most severe risks are dependent on that the attacker can get physical access to the device, which should be possible to prevent if the user acts in a cautious way.

These risks and their severity are fairly well weighted between the two systems. Some attacks are easier to perform on iOS and some are easier with BlackBerry OS, but the fact that they exist in both systems is still important.

The many identified risks are showing that both systems are in need for hardening in some extent to be safe to use. For companies that work in such fields which require confidentiality it is really important for their IT-departments to specify policy rules and restrictions in order to ensure the company confidentiality.

The most interesting risks are in some senses the GSM attacks. These are really hard to do anything about, since GSM is old and broken and cannot provide the confidentiality and integrity that is necessary today. Especially is this becoming a bigger risk as a result of the open source projects and cheap hardware that has entered the market making it possible for pretty much anyone to buy.

In the next chapter are solutions and mitigations for these identified risks given to the extent that it is possible to resolve these risks.

8 Solutions and mitigations

In the previous chapter a lot of risks were identified which are valid for both iOS and BlackBerry OS, and these risks make use of such devices insecure. In this chapter are solutions and mitigations for the identified risk given to show how to strengthen these systems and make them more secure. How to prevent the identified attacks in the previous chapter is discussed here to show how the risk can be mitigated by preventing these attacks. Some recommendations are directions for device configurations and others are suggestions on how to make the user more aware about the possible risk. It is however important to understand that these solutions and mitigations do not make the system entirely secure. Hence, this is not all that must be done for achieving a reasonably secure system.

8.1 Jailbreaking/Rooting

Jailbreaking/rooting is done by exploiting an implementation bug found in the system and it gives full root access to the device. This is quite hard to prevent for the regular user since neither iOS nor BlackBerry OS are open source systems, so a regular user cannot fix bugs by himself (and it is probably too complex for the average user anyway). As stated earlier there is no way to root BlackBerry OS 6 devices known at the moment, so for BlackBerry users this risk is not big enough to care about, but for iOS there are a few different known ways to jailbreak devices. However, Apple releases now and then updates for iOS and usually the new versions fix these implementation bugs that previous jailbreaks depend on. So, it is important for users to always update to the latest versions and continuously update their devices and applications.

Besides the technical way to prevent jailbreaking there also are things that the user can do to make it harder for an attacker. Jailbreaking requires physical access to the devices, so the best thing a user can do to stay safe is to never let unauthorized persons access the device. This includes not leaving it unattended in places where other persons can access it, e.g. public locations, hotel rooms, unlocked offices and so on. A user should never lend the device to people that he does not know, not even to a beautiful person at the pub that just wants to call a cab. It is really important to restrict the access of the device to a minimum of only trusted persons in order to minimize the risk.

Some users jailbreak their device themselves to gain the possibility to use unsigned applications and to get access to otherwise restricted parts of the system. This is absolutely not recommended, since it removes almost every security aspect of the system, and it is a bit like asking to be a victim.

8.2 Malware attacks

Malware can quite easily be implemented and there is no way to prevent that from happening. There are a lot of different anti-malware softwares and other tools that help to protect regular computers from malicious softwares, but this is not the case for smartphones. There are a few products that look for malwares for BlackBerry OS, but not to the same extent as for computers. Most such applications only check the checksum of the installation package to see if it is a previously reported malware. These applications would therefore not recognize a newly written and specific malware, such as the one created for this thesis work, before it has been reported. For iOS there are no anti-malware applications and the existing ones for BlackBerry OS are not enough in order to protect a device from malware. Hence, it is up to the user to ensure that the applications that are installed are free from malware.

The best way to avoid getting malware is to only install applications that come from reliable and trusted sources. For BlackBerry OS devices it is recommended to only install applications directly from RIM's App World and for iOS devices directly from Apple's App Store. Even though this does not guarantee that the application is malware free, the risk is very much reduced. The user should also think about what applications he installs and it is recommended that only necessary applications are installed. So, it is probably a good idea to skip games and such applications.

8 Solutions and mitigations

For business usage it is recommended that the corporate IT-department sets up policies and rules for smartphones and makes restrictions on the devices so that rules must be followed. For the case with malware IT-departments can restrict what applications can be installed and preferably also prevent the user from installing applications, so only the IT-department can install them. For iOS devices this can be achieved by using configuration profiles (see section 3.3 for details). In the category *Restrictions* such application restrictions can be configured. For BlackBerry OS devices using BES a lot of application restrictions can be set and pushed out to devices; possible restrictions such as: only accepting applications from App World, only accepting applications included in a specified list of applications and so on. To learn how to set up such restrictions for BlackBerry OS the user is encouraged to consult [27].

8.3 Phishing attacks

Phishing attacks are successful to a much higher degree on smartphones than on computers [47] and this fact is important to consider. Both iOS and BlackBerry OS are vulnerable to these attacks and both actually have behavior that makes it easier for an attacker to perform such attacks. Safari, the default web browser for iOS devices, has the feature that the URL field can be hidden when browsing to large web sites so the device can show as much as possible of the web site. This is done by adding a few lines of code in a JavaScript that runs when the web site is loaded. JavaScript can be used with a BlackBerry OS device's web browser as well to hide and modify information. This behavior makes it easier for an attacker to hide the fact that a web site is a phishing site. So, to mitigate phishing site attacks it is recommended to disable JavaScripts in the system. For iOS devices this is done by going to *Settings->Safari* and turn off *JavaScript* and for BlackBerry OS 6 devices this is done by going to *Browser->Options* and uncheck *Enable JavaScript*.

However, even without JavaScripts phishing sites can be made, and other phishing attacks, such as phishing emails, can still occur, so the system is not entirely protected from this kind of attack. Actually, disabling JavaScripts only makes it easier to notice phishing behavior; it does not prevent the attack. So, it is still up to the user to be cautious when entering confidential and sensitive information on the Internet and in emails.

8.4 Attacks on encryption

As stated in 7.5.4 the encryption algorithm used in both systems is secure enough to protect the data stored on devices if the key management is secure enough. For BlackBerry OS devices the key generation and storage are highly secure and no bypassing of the encryption protection has so far been publicly showed. However, the entire key hierarchy depends in the end on the ephemeral key that is generated from the device password chosen by the user, and if this password is weak, brute force or other password cracking algorithms is possible. Therefore, it is really important to have a strong password for the device, and it is recommended that password rules are implemented and managed by the corporate IT-department for business usage. A strong password should be enough to secure BlackBerry OS from this type of attack. A strong password should be at least eight characters long and contain uppercase letters, lowercase letters, digits and symbols. By using enterprise policies administrators can force the use of strong passwords for the device. To learn how to construct such rules, consult [27].

The key management for iOS contains flaws that make the device encryption vulnerable to attacks [9]. These attacks use the possibility of jailbreaking, so it is important to follow the suggestions given in section 8.1 above to make it harder to perform such attacks. However, if a weak device password is used, it is also possible to brute force the password and in this way crack the encryption. So, it is strongly recommended, just as for BlackBerry OS, to use a strong password and not only the four-digit passcode option for iOS. By using configuration profiles IT-departments can force the use of secure passwords instead of weak passcodes for devices and it is strongly recommended to do so.

8.5 Attacks on 3G/GSM

The usage of 3G and GSM networks is common for more or less all smartphones on the market today. GSM

is old and it contains flaws, which makes it too insecure to use for sensitive and confidential calls and messages. 3G are still considered to be secure enough but it is easy to block 3G signals and in this way make today's devices switch over to GSM. It is important to understand that this vulnerability exists with more or less all devices today, hence it is not only a problem associated with iOS and BlackBerry OS. There are only two things to do in order to prevent and mitigate these kinds of attacks. The first way is simply to not use GSM at all, but with the current available devices it is not possible to turn off GSM (actually it is only possible to turn off 3G), so it is once again up to the user to continuously check the connection when making calls.

The second way is to use voice encryption hardware/software so the conversation is encrypted and decrypted at the device so that the conversation never enters the network in plaintext. This solution is more secure but it has its disadvantages since such tools are quite expensive and it requires the callee to use the same tool, which decreases the amount of callees to call securely. Another thing to think about with this method is that these tools are developed by third parties and it means that one more party must be included in the trust chain for the device security. However, Kryptos is a voice encryption application that works both with iOS and BlackBerry OS devices and is available both on Apple's App Store and RIM's App World. Kryptos is mostly targeting the commercial market for businesses but even private users may buy their solution. More information about Kryptos can be found at [53].

8.6 Hardware tampering

Hardware tampering does require physical access to the device and the best prevention for this attack and the associated risks is to never leave the device unattended at anytime, as discussed in section 8.1 above. Since tampering modifies the device it is also possible to check for differences and marks which, indicate that something is changed. For instance, with BlackBerrys is the battery the most probable part to exchange since it is easy to remove. The user should therefore check that the battery is unmodified, if the device has been left unattended. The easiest way to check the battery is to note the battery serial number, since each battery has an individual number printed on them. The probability that the attacker knows the victim's number is low and if the battery is exchanged this can probably be noted from this. iPhone is a solid piece that must be demounted using a screwdriver, which makes it harder for an attacker to tamper with it, but also harder for the user to control the hardware.

Another way to make the devices a bit more tamperproof is to use tamper-sealing labels that can be specifically designed for the user. These labels are small adhesive tags that can be placed on devices so that they cannot be opened and modified without being noticed. These labels also make it possible to notice if the device itself is exchanged for another device. Such labels can be user designed and ordered from web sites on the Internet, e.g. [54].

The above recommendations do not make it impossible to tamper with a device, it only makes it easier to note if a device has been tampered with and it would reduce the damages made from this attack. The only way to protect a device from tampering is to never leave the device unattended.

8.7 Spoofed Wi-Fi access points

Both iOS and BlackBerry OS by default accept usage of insecure and unprotected Wi-Fi networks. This leaves the decision about using an available network or not to the user. Average users value usability higher than security, so in many cases the user will probably accept public available Wi-Fi networks. Therefore, it is important, especially for business usage, to restrict this feature in order to ensure the security for the device and the data transferred from the device. For iOS, this can be restricted by using configuration profiles. The category *Wi-Fi* contains all restrictions that can be made. The minimum if this restriction is turned on is to state the SSID:s of all trusted Wi-Fi networks. This is however not enough, since an insecure Wi-Fi network can be named with the same SSID as a trusted network. So, it is recommended to not only enter the SSID, but at least enter the security type that the network should require as well.

For BlackBerrys using BES can this restriction be made using policies and be pushed out from the BES

server. There are much more possible restrictions to be made with BlackBerry than with iPhone. For instance can untrusted Wi-Fi networks be blacklisted, any network can be used that fulfills a required encryption level and so on. Consult [27] for more information on Wi-Fi restrictions for BlackBerry OS.

8.8 Network attacks

Network attacks are in most cases depending on implementation bugs and/or misconfigurations of the device and it is therefore hard for a user to do anything to prevent them other than regularly and continuously update their device to the latest version. Usually both Apple and RIM release patches quite fast that fix known vulnerabilities in their systems, and it is therefore important to update as fast as possible to minimize the time a device is vulnerable for an attack exploiting a specific vulnerability.

As stated before both systems are using the browser engine WebKit, which has been identified many times before to be leading to serious vulnerabilities in both systems. At the moment there are no web browsers for iOS and BlackBerry OS that do not use WebKit. The processing of JavaScripts in WebKit has been faulty and it has been the issue for many of the earlier fixed vulnerabilities in WebKit. There is at the moment one unfixed vulnerability for BlackBerry OS issued by JavaScripts in their web browser [55]. A similar vulnerability did exist in iOS but it has been fixed in the latest version. However, it is recommended to disable JavaScript in both systems to prevent the current vulnerability as well as protect them from any future vulnerability depending on JavaScripts. How to disable JavaScripts in these systems is described in section 8.3 above, but for BlackBerry this can be forced to the device by using policies as well, see *Browser IT policy* rule in [27].

8.9 Theft

Theft can be performed in many different ways including burglary, robbery and pickpocketing. It is, once again, important for the user not to leave a device unattended at any time and always treat the device with caution to make it harder for a thief. However, theft can happen even with the most cautious user, for example by armed robbery, where it is not recommended to risk the life for a device, so it is important to mitigate the damage made from a theft as much as possible.

The first recommended thing is to ensure that the user has all data backed up, so that loss of the device does not lead to loss of all data as well. One important thing to remember is that a backup of a device usually means that all data on the device is copied to a computer or an external disk. Hence, this backup is important to protect as well. So, it is recommended that backups are encrypted and stored in a safe way. The second thing to think about is to not make it easier than necessary for the thief to access the data on the device, so always use strong passwords and make sure that the device is being locked automatically. The last recommendation is to activate the possibility of remote wipe and device tracing. For BlackBerrys, this is done by sending a remote signal from the BES server and for iOS it is done by creating a MobileMe-account and installing the free application Find My iPhone [14].

An important thing to understand is that the remote wipe feature is quite easy to stop since the wipe signal is issued over the mobile phone network. Hence, by removing the SIM-card or by in any other way making the device lose the network connection this feature is dismantled. So, a thief that is interested in the data on the device will probably get it anyway. It is however recommended to try to wipe the device if it is lost.

9 Hardware token

The increasing use of smartphones and the degree of sensitive information stored on such devices makes the need for a secure platform indispensable today. As pointed out in this paper is neither iOS nor BlackBerry OS secure enough to in any greater extent be trusted with sensitive and confidential information. Hence, some kind of platform hardening is eminently needed for strengthening these platforms.

Besides the software hardening described in the previous chapter it is desirable to use a dedicated hardware token to strengthen these platforms. As stated in section 1.2 one aim of this thesis is to come up with a suggestion on how such a token could be utilized and designed in order to strengthen iOS and BlackBerry OS. This chapter is presenting and discussing the possibility of such token, including the dream scenario, problems, design and authentication with this hardware token.

The dream scenario is describing how it is desirable to work in the best of worlds. This is actually not a utopia, but it is not possible to implement into these two systems today. Therefore, the shortcomings are discussed and a more realistic version of the hardware token is discussed in order to be valid for these two systems today. Sadly no sufficient token suggestion is given and the reasons of this are therefore discussed.

Authentication is discussed since it is desirable that the hardware token also can be used for authenticating a user for the system. Authentication with a hardware token leads to another more secure possibility for locking devices. Instead of only requiring a password to unlock the device the token can be combined with a password/PIN required to unlock the device. This two-factor authentication will lead to a more secure device lock.

The following sections of this chapter are describing the hardware token in full and all problems with implementing such a token in these systems are discussed. At the end of this chapter has some conclusions regarding hardware tokens been summarized.

9.1 Dream scenario

In the best case would the same hardware token be valid regardless of the system it is used with, i.e. the same token could be used with BlackBerry OS, iOS, Android OS, as well as regular computers with their operating systems and so on. This token should be able to authenticate the user in a safe and secure way and also prevent all possible risks on these systems. In the real case this is really hard to accomplish, since these systems have huge differences and are also exposed to different risks. However, this section is discussing a dream scenario of such a hardware token in order to present what to strive for.

9.1.1 Dream utilization

The token is desirable to be small enough to be easy for the user to carry with him and should be possible to connect to all possible systems. Some smartphones, as well as all computers, can be used as USB-hosts and therefore a possible solution is to connect the token by USB to the device. The dream case is that when the token is connected to a device and a password is entered to unlock the token, a safe and secure virtual platform is mounted on the device, totally separated from the underlying platform. All sensitive data is stored encrypted on the token and all conversations are encrypted in the virtual platform and only use the underlying platform as a modem that sends the encrypted bit stream. In this way most deficiencies in the underlying platform are irrelevant, since the underlying platform only can access the encrypted data and nothing else. This leads to that no specific phone is needed at all since no sensitive information is stored on the phone itself. This makes the device platform less important so mainly the token is important to protect. Since USB is used the token can be reused with any system that manages USB connections, typically a computer system. To authenticate the user it should be enough to use a strong password, which will lead to two-factor authentication since both the token and the corresponding password is needed.

This token is desirable since it reduces the need for hardening the underlying platform, since the virtual

9 Hardware token

platform is shielded from the underlying platform and is therefore protected from most shortcomings of the underlying platform. One thing that is hard to work against is the fact that the password is entered at the device before the virtual platform is mounted, and there is therefore a risk that malware can sniff the token password when it is entered.

Since the token contains all sensitive information encrypted and the token itself has no connection to the internet or mobile network the possible attack time is reduced to only the time the token is connected, which should be much less time compared with the phone that often is connected to a network twenty-four hours a day.

9.1.2 Platform limitations

The above described dream scenario is not impossible to happen in a few years, but with today's existing platforms, such as iOS and BlackBerry OS, limitations exists that makes this utilization unlikely to be possible to implement today. First of all are iOS devices and BlackBerry OS devices unable to work as USB-hosts, which are needed for this to work. This is coming on many new models of smartphones, and it is probably only a question about time before all devices support this, including iOS and BlackBerry OS devices.

Another problem is the big difference in the architecture between these platforms, which is hard to work against. It makes it hard to mount a shielded virtual platform, since the hardware as well as the operating system differs which requires the token to be compatible with many possible platforms.

iOS has also very hard restrictions on what can be mounted on their devices and at the moment it is not possible to mount a virtual platform at all on iOS devices. This restriction is not possible to bypass without clearance from Apple or without jailbreaking the device, which is not a solution. It is very unlikely that Apple will allow and enable this feature in their devices in the near future.

iOS devices do not follow the accepted standard of using micro-USB as a mobile charger connection [56] (which is also used for USB connection to the device). This means that a universal token for both iOS and BlackBerry OS devices is not possible without an extra adapter for iOS devices. BlackBerry does follow the recommended standard for micro-USB connections and it is also much less restricted for mounting a virtual platform on the device, since it is built on a version of a Java Virtual Machine. However the lack of USB-host hardware is still making it impossible.

9.2 Token suggestion

Since both iOS and BlackBerry OS lack the capability to be used as USB-hosts the dream token cannot be implemented. Therefore, other designs of hardware tokens are discussed in this section. First is the design and usage discussed and then is the authentication method described and last is the reusability between platforms discussed. The wanted capability that the token should be able to be used with almost all possible platforms has been dropped since it is unreasonable to implement such a token in the real world today. However, usage for iOS, BlackBerry OS and a computer system is discussed below.

9.2.1 Utilization

The hardware token has some requirements on what strengthening it is supposed to introduce to these systems. First of all, it should introduce a secure way for authenticating the user so that only a user with the right permissions can access the device. It should also strengthen the malware protection and detection possibilities for the platform. It should protect stored data from disclosure to an attacker that has physical access or remote access to the device. The last thing it should introduce is protection for eavesdropping on calls and data traffic. All these requests are hard to meet with one token and it might therefore be hard to introduce all of them in a satisfying way, especially if the request for reusability between platforms should be met. The authentication part as well as the reusability part for this token is described below.

To harden the security of data stored on the device the data is required to be encrypted in a more secure way and the keys must be managed in a safe way. A good solution would be to have the entire device encrypted using keys generated and securely stored on the token. So, when the token is connected and the user is authenticated the token can decrypt data that should be used. The problem with this is that no third party applications can access the entire memory system on the device. They are only allowed to access their own small part of the memory assigned to their sandbox, which means that a third party solution cannot encrypt the entire device in neither iOS nor BlackBerry OS. A less efficient solution can be to implement an application that replaces the device's address book, calendar and notes and so on, with its own version of these features. Hence, the application can securely encrypt all this data and require the token to be connected to be able to decrypt it. This would make some of the sensitive data more secure, but it would not secure all sensitive data since for instance emails, photos and keyboard cache will not be possible to access from such an application and therefore these will not be protected.

To prevent calls and data from being intercepted a token cannot do much under the current circumstances, since the baseband module is not accessible for third parties, which means that restrictions for usage of the GSM network is not possible neither in iOS nor BlackBerry OS. However, a third party application can implement VoIP-calls in a secure manner. So, if VoIP can be used instead of regular phone calls the application can use the token to encrypt the outgoing bit stream before leaving the device. This would not prevent interception of the call, but at least make it impossible for the eavesdropper to get the plaintext. VoIP clients are indeed possible to implement for both systems, in fact there already exist a bunch of them on the market, e.g. Truphone [57].

Malware detection and protection is a well-established problem with these two systems since there are very few tools available for BlackBerry OS and none for iOS that introduce this to these systems. One of the reasons for this is, once again, that third party applications do not have access to all files in the file system and can therefore not run an anti-virus scan on the entire device, just in its own memory space. This is a huge problem for implementing any secure and reliable malware detection and it is not easy to do anything about this. The best thing is instead to work with malware prevention, that is preventing malware from entering the system instead of detecting and neutralizing them when they are installed on the device. Apple and RIM are doing as well as they can with whitelisting and blacklisting applications that are known to be malicious, but as shown this is not always enough. What the token could contribute with is a specific list of blacklisted applications, which will be used to check that none of these applications is installed on the device when it is connected. However, the token cannot do anything else than notifying the user about this possible threat, since it does not have the permissions needed for removing applications. This does not insert much security and the list must continuously be updated with new lists of malicious applications.

Regarding the physical design of the token a smart card is the most appropriate thing to use. Some BlackBerrys already have a smart card reader built in, and for those models that do not have it, and for iPhone, an external smart card reader can be used. There are a few versions for smart card readers available for these systems, e.g. BlackBerry's own Bluetooth smart card reader [58].

9.2.2 Authentication

The token should be used for authenticating the user as well as hardening the security of the platform. The dream scenario where all sensitive data is stored on the token is not possible at this moment and therefore the device will still contain sensitive data. Hence, it is important that only the correct user may access the device. It may also be of interest to know that the device is the correct one. The authentication method that seems to be appropriate for this is that the token contains its own private key together with the device public key. The device should contain the opposite, i.e. its own private key and the public key of the token. The suggested authentication method is a kind of challenge-response authentication and it should work as follows. When the token gets connected to the device it asks the user to enter a password to enable the token. When it is done, the device randomly generates a string working as the challenge and the device encrypts it with its own private key and then encrypts it with the token's public key before sending it to the token.

The token first decrypts it using its own private key and since it requires the private key of the token, only

the correct token will be able to do this. The token is then decrypting it again but now using the device's public key, which means that the token can be certain that this challenge was sent from the specific device. When the token has the challenge string in plaintext, the token will encrypt it using its own private key and then using the device's public key before returning it to the device. The device will then do the same as the token just did but by using the corresponding keys. In this way the device will get the original challenge string back and both the device and the token knows that it communicates with the correct and trusted part.

This authentication method may seem a bit complex but it has its advantages. First of all will this method both authenticate the token (and thereby also the user) and the device, which can indeed be of value since it makes it much harder for an attacker to replace the user's device without notifying the user. Second, it does provide a two-factor authentication together with a challenge-response technique. The challenge-response technique prevents the possibility of playback-attacks, since the challenge will be different each time, which leads to that a previous used challenge will not grant access to the device the next time [1]. The two-factor authentication is achieved since the user must have both the token and know the token's password. The need for both factors makes it harder for an attacker, since it is harder to retrieve both than only one of them.

Instead of using a password as the second factor for the token biometrics can be used in the two-factor authentication. A possible solution could be to have a fingerprint reader on the token or perhaps having face recognition software on the device using the device camera. It can be convenient for the user to use biometrics instead of remembering a strong password. However, biometrics has still some problems with either false positives or false negatives depending on the configuration [1].

Another advantage is that this authentication method is possible to implement in a similar way on iOS and BlackBerry OS as well as a computer system, which is a desirable capability. A disadvantage is that the token and the device must be configured and initialized at the first time of use since the keys must be exchanged and the password must be chosen. This is however a fact that is very hard to evade.

9.2.3 Reusability

The reusability of a token is depending on the differences in the platforms the token is supposed to work with. In this case it is mainly iOS, BlackBerry OS and computer platforms, such as Windows and Mac OS versions. It is of course also depending on the support for the token interface with the platform, e.g. USB is not supported for all these platforms. The differences between iOS and BlackBerry OS are big and it makes it hard to implement a token that can be used with both of them. For instance are applications executed in iOS running in the Apple Sandbox, which is a C-based sandbox and the applications are objective-C based, while with BlackBerry OS applications are running in a Java Virtual Machine and the applications are Java based. So, it is not possible to have one application for the token, it must have separately created applications. This is however, a small thing compared with the differences in the kernel that make the token possible to connect.

The reusability with a computer system is probably more conceivable to achieve and it depends on that most computer systems today are quite adaptable and can be configured to a much greater extent than smartphones may.

9.3 Conclusions

Both iOS and BlackBerry OS are tightly closed platforms that leave almost no possibilities for a third party to modify anything in the core. This can of course be a good thing since it makes it hard for an attacker to change things in the core, but in this case, when trying to design a hardware token in order to strengthening this platforms it is a bad thing. It limits the possibilities for what can be done considerably and as a result of that a useful token cannot be implemented in a satisfying manner. The opposite of this is Google's Android OS that is open source and a third party can rebuild the core as much as they want. Android OS does however suffer from other things that in some sense make it much more insecure than iOS and BlackBerry OS.

Regarding the reusability for a token between these platforms it is also limited what can be done. At the moment are the differences in the platform structure so big that it is really hard to design one token that can be used for both of them. For instance, the smartphone will need an application that handles the token connection and this application must be specifically created for each platform since applications for one system cannot run on the other system.

To summarize the conclusions for the hardware token; it is not possible to implement such a hardware token that can ensure smartphone integrity for these systems, and especially not a reusable token for both systems. The only thing that can be implemented with a token in a sufficient way is the authentication part and that is because both systems have a built in hook letting developers implement own ways to authenticate and unlock these systems. However, it is probably not justifiable to implement a hardware token only for the authentication part of the smartphone, since it probably costs more than it achieves.

10 Results and discussion

In this chapter is the result of this thesis work presented and discussed. First is the result for each part of the work enlightened and described and then are conclusions that have been drawn from the result discussed and motivated. The outcome of this work is also discussed and motivations for some of the used methods are given. Last of all are some recommendations for future work stated and it is motivated why these fields are worth investigating in greater detail.

10.1 Results

This thesis work can be said to consist of four major parts and all four have been important for the outcome of this work. The first part was platform research and it resulted in an important understanding of how each platform is working, which features they have and how these platforms are being used. This part is described in chapters 3 and 4 in detail. The information and knowledge gained from this part was indeed important for the rest of the work since it was the fundamental knowledge required to be able to analyze these systems in a reliable way.

The second part was the practical testing in the form of the proof-of-concept malware and the GSM-attacks, which was intended to investigate and demonstrate that these types of attacks are possible and actually quite easy to perform. The malware test resulted in a proof-of-concept spyware that collects sensitive data from the device. For instance, it collects call history, the keyboard cache and it records audio from its surroundings. The proof-of-concept spyware is described in chapter 5 in greater detail. The GSM-attacks that were tested were using a rogue BTS to trick victims' devices to connect to it. It led to that all calls made and SMS that were sent from a victim's device could be intercepted. This test resulted in a configured BTS working in a malicious way, and it could, without problem, eavesdrop on calls and it could intercept SMS, regardless of the platform the device was running. This attack is described in chapter 6 in detail. The result of both these tests shows clearly that there are ways to attack the integrity of both iOS and BlackBerry OS devices as well as their users' privacy.

The third part was to perform a risk analysis of each of these platforms in order to evaluate if these systems are secure enough for today's use-cases. Almost all larger companies have at least a few employees that use smartphones in their private life and/or at their work and therefore are the requirements for secure devices growing stronger every day. This part resulted in that not less than 19 risks were identified together with the corresponding vulnerabilities that enable these risks. Many of these risks seem to only occur when an attacker can get physical access to the device and therefore must a device be kept under supervision all the time. The risk analysis is described in chapter 7 and in Appendix D: Risk tables, is a risk table shown for each identified risk. Some of these risks are very severe and must be considered by all persons using smartphones with these operating systems.

The fourth and last part of this thesis work was about giving solutions on how to mitigate and/or prevent the identified risks and strengthening these platforms in general. It resulted in a bunch of configurations and usage recommendations that mitigate many of the identified risks. Some of them are making the attack impossible or at least more complex to perform and others just reduce the impact if an attack succeeds. An important point in this part was also to investigate the possibility of using a separate hardware token to strengthen these two platforms. The result there is, to some extent, a disappointment since the suggested hardware token would not strengthen these systems as much as needed, and the token must be implemented in different ways for each of these system and this was not the desired result. The mitigation recommendations and the hardware token are discussed in chapter 8 and 9 respectively.

10.2 Discussion and conclusions

What conclusions can be drawn from the result of this thesis work? First of all, both iOS and BlackBerry OS have flaws and vulnerabilities that make them insecure and which can compromise company security as well

10 Results and discussion

as user privacy. For instance are both systems sensitive for malware which can spy on the user as well as lead to financial loss for the user. These risks exist since no sufficient malware protection exists for these systems. iOS and BlackBerry OS are also lacking security frameworks to ensure the integrity of the device, and a few of the existing security features they have are insecure and only leads to false security confidence. A good example of deprecated security in iOS is the encryption key management, which states that the encryption is strong but in fact the encryption key can be accessed if the attacker can get physical access to the device. BlackBerry OS has also important security flaws, for instance it is lacking data execution prevention and address space layout randomization, which makes it much easier to exploit implementation flaws on their devices.

Perhaps the most eye-opening result was how easy it is to set up a rogue BTS which can be used for eavesdropping on victims' calls. It is not news that this is possible to do, law enforcement has used this thing for years, but the equipment is then called an IMSI-catcher. The news here is how easily, fast and relative cheaply it actually can be done today. Starting out from scratch without any knowledge about this kind of thing to having a fully functional IMSI-catcher took only a few workdays and cost less than \$1500. Even though this IMSI-catcher has a short range, it is a very interesting and scary attack. Perhaps the scariest part of this attack is that it is possible to attack almost all mobile phones on the market today. It is also hard to do anything about this attack, since it is based on flaws in the GSM-network, but what easily can be done by the device manufacturer is to actually acknowledge the network's warning that a call is made unencrypted instead of just ignore this warning. Then will the users at least be notified about this and can choose to not make the call under such circumstances.

Many of the most severe risks do require the attacker to get physical access to the device and others are using social engineering attacks to trick the user to perform the attack himself. So, one of the most important things with these systems to ensure phone integrity is to be a smart user! Much of the security within these systems does lie in the user's hands, so if the user acts in a smart and secure way will many of these risks be mitigated. It is therefore very important for companies to have clear rules and policies on how smartphones may be used inside the company and it is equally important to train employees in smartphone security and behavior, as it is important to train them in computer security. Well-formed rules and preferably also using configuration profiles for iOS and policy restrictions for BlackBerry OS to ensure that the users must follow the rules is highly recommended for business usage.

Another important thing when thinking about security is to evaluate the chain of trust for the system. There are lots of parties that must be trusted when using a smartphone, and if only one of them is behaving in a malicious way the entire phone integrity can be compromised. For instance a user of these systems must trust that Apple/RIM has not placed malware, microphones and/or other spyware tools on the device from the beginning. A user must also trust the mobile network carrier to not eavesdrop on calls as well as trusting the application writers and so on. So, for companies, authorities, military and governments that treat highly sensitive and classified information it is extremely important to evaluate if all parties can and should be trusted.

A bit of failure and disappointment is the work with the suggestion of a hardware token in order to strengthen the security of these systems. All investigations only resulted in identified problems with all ideas for a token, and the major reason for this is the tight restriction for third party modification in these systems. Especially with iOS it is almost impossible to do any platform hardening without jailbreaking the device, which by itself makes the system less secure. The few suggestions given should however be possible to implement and they should at least result in a bit more secure platforms. The authentication part is indeed possible to implement, and it is a step in the right direction compared with the original authentication method.

To summarize, do not put too much belief into these systems and always be a smart user when using smartphones! Remember, smartphones are not just simple mobile phones, they are devices with power as a small laptop and contain at least as much sensitive information as a laptop does, so treat them with at least the same caution that you treat your laptop!

10.3 Future work

The aim of this master thesis was to test and evaluate the security and integrity of the two smartphone platforms iOS and BlackBerry OS and then present solutions and mitigations on how to strengthening the security of them. The smartphone market is one of the fastest growing markets today and the performance of today's smartphones is increasing for almost every day. New versions of these platforms are coming with new features and new use cases and with them will of course new flaws come that lead to new attack possibilities as well as new risks. So, risk management is a forever-ongoing work in order to stay on top of system security; and smartphone systems are no exception! Hence, the work made in this thesis must go on in order to continue to achieve smartphone integrity for these systems.

The implemented attack with the rogue BTS is probably only a small part of what actually can be done. Sadly was this part added to this master thesis quite late in the process, so not as much time as wanted could be spent on this part. GSM is still widely used, and not only by these two platforms, so this kind of attack is indeed interesting to have a closer look at. So, a recommendation for future work is to have a closer look at this, including looking at Over-The-Air Programming by SMS in greater detail than what has been done in this thesis.

The suggestion of a hardware token that is presented in this thesis can preferably be investigated in greater detail. Tentatively should practical testing of ideas be made in order to see exactly what can be done, how complex it is and how it can be improved, since it has only been a theoretic study in this thesis. There probably are a lot of other solutions for hardware tokens that can be used together with these platforms and therefore it can be of interest to study and perhaps test some other solutions as well.

Smartphone integrity is a very hot subject today and there exist a lot of things to test, evaluate and figure out fixes for. The depth of the made risk analysis can for some purposes be too thorough and for other cases can it be partial, and in further analysis it is recommended for the analyzer to adjust the depth according to the need for its specific case.

11 Definitions and abbreviations

3G:	Abbreviation for 3:rd Generation and it is a standard for mobile phone communication systems. It has much higher transmission rates than GSM and it has improved security features to make eavesdropping harder [32].
AES:	Abbreviation for Advanced Encryption Standard and it is a symmetric encryption algorithm that is using keys of size 128, 192 or 256 bits. It is considered to be very secure [59].
API:	Abbreviation for Application Programming Interface and it is an interface for software, which grants access to services and functionalities that is provided by other softwares or libraries. This is done so that services can be reused and easily be accessed [60].
APN:	Abbreviation for Access Point Name and it is a protocol used for granting access to internet over the mobile phone network.
App:	Abbreviation for Application, which is the name used for softwares in smartphone systems.
ARC4:	Is also known as RC4 and it is a software stream cipher. The algorithm generates a bit stream of data of the same length as the plaintext. The bit stream is generated from the encryption key. The encryption and decryption is then a bitwise XOR of the plaintext and the bit stream [59].
ASLR:	Abbreviation for Address Space Layout Randomization and it is a security feature that makes the address space for a process to move around randomly in the memory space each time it is started. It makes it much harder for attackers to attack the system [13].
Backdoor:	A backdoor is an open way into a system that is used for bypassing security features of a system. For instance, the authentication process is often bypassed. Backdoors are generally opened by an attacker so that he later easily can access the system [61].
Baseband:	A baseband is the radio module chipset and the module firmware in a device. This module handles all mobile phone radio signals (GSM, 3G and so on), the Wi-Fi signals and the GPS signals in smartphones.
BES:	Abbreviation for BlackBerry Enterprise Server and it is one of two possible usage modes with BlackBerry devices (see BIS for the other). BES is mostly targeting enterprise users and it has a much higher security than BIS but it requires an enterprise server within the company [22].
BIS:	Abbreviation for BlackBerry Internet Service and it is one of two possible usage modes with BlackBerry devices (see BES for the other). BIS is mostly targeting regular users and it provides internet access for the device through RIM's own BIS servers [21].
Bootloader:	The bootloader is a piece of code that is executed at the very beginning of the start up of a device. The bootloader contains all data needed for initializing and starting the operating system and it loads the operating system into the memory [62].
BTS:	Abbreviation for Base Transceiver Station and it is the transmitting and receiving part of a GSM base station.

11 Definitions and abbreviations

CAST5:	CAST5 is a symmetric encryption algorithm. The name is made up from the initials of the inventors, Carlisle Adams and Stafford Tavares [59].
CPU:	Abbreviation for Central Processing Unit and it is the processor in a computer that executes all instructions the computer should perform [62].
CSP:	Abbreviation for Carrier Service Provider and it is a company that provides access to mobile phone services for phone subscribers.
DSA PRNG:	Abbreviation for Digital Signature Algorithm PseudoRandom Number Generator and it was intentionally implemented to generate pseudorandom numbers to be used as seeds for the digital signature algorithm. The output from this algorithm is a 64-byte large pseudorandom number [26].
ECC:	Abbreviation for Elliptic Curve Cryptography, which is an asymmetric encryption algorithm. The algorithm is based on the mathematics of elliptic curves and finite fields. One of the benefits of ECC compared with other asymmetric algorithms is that the key length can be significantly shorter [25].
EDGE:	Abbreviation for Enhanced Data rates for GSM Evolution and it is a further development of GSM with higher transmission rates than GSM [32].
GPRS:	Abbreviation for General Packet Radio Service and it is a mobile data service based on the GSM system for mobile communication. GPRS is providing internet access without requiring the modem to dial-up before use. It was this service that made MMS possible [32].
GPS:	Abbreviation for Global Positioning System and it is a satellite system used for navigation.
GPU:	Abbreviation for Graphical Processing Unit and it is a processor that is specifically made to be able to perform graphical calculations very efficiently [62].
GSM:	Abbreviation for Global System for Mobile communication and it is a standard for mobile phone communication systems. It is the most common mobile conversation network and most of today's mobile phone calls are still using this system [32].
ICCID:	Abbreviation for Integrated Circuit Card IDentifier and it is the SIM-card's international identification number. This id is among other things telling which carrier the device is using and the country the device is registered in [1].
IMEI:	Abbreviation for International Mobile Equipment Identity and it is a unique identifier for a device that is using the GSM network. This identification number is used in GSM for identifying valid devices in the network [1].
IMSI:	Abbreviation for International Mobile Subscriber Identity and it is the unique identification number of a SIM-card inside carriers' own networks. This is used so that the carrier can establish which SIM-card a phone call is made from and much more [1].
IPC:	Abbreviation for Inter-Process Communication and it is a collection of methods for enabling communication between different threads in a system. This also provides the possibility of sending data between threads in a system [62].
IPSec:	Abbreviation for Internet Protocol Security and it is a bunch of protocols used for making

IP-communication secure by encrypting each sent packet in a session [63].

- ISP:** Abbreviation for Internet Service Provider and it is a company that provides access to internet for their customers.
- J2ME:** Abbreviation for Java 2 Platform Micro Edition and it is a Java platform mostly used for mobile devices [60].
- Jailbreaking:** Jailbreaking is the process of using some known implementation vulnerability in iOS to gain root access for a device. With root access can all capabilities in the system be unlocked and user limitations can be bypassed. This also makes it possible to bypass almost all security features implemented in iOS, e.g. whitelisting of applications.
- Java reflection:** Reflection is when software is allowed to modify the structure of itself or of other source code at runtime. This makes the software dynamically changeable and this can among other things be used for mapping objects with sources at runtime [60].
- JNI:** Abbreviation for Java Native Interface and it is an interface that makes it possible for Java code executed in a JVM to be called from native softwares and from the operating system [60].
- JVM:** Abbreviation for Java Virtual Machine and it is an isolated runtime environment where Java applications are executed. The JVM is working as a physical machine but it is just a software implementation of a physical machine [60].
- Keylogger:** A keylogger is a piece of malicious code that is logging everything written on the keyboard. It is considered as a type of malware and is often used to collect sensitive information such as passwords and financial credits [61].
- L2TP:** Abbreviation for Layer 2 Tunneling Protocol and it is a network protocol used by VPN to open a direct tunnel into a private network. The protocol is not providing any security for data transferred in the tunnel so extra protocols are often used [63].
- Malware:** Malware is the collection name for all types of malicious code that can affect computer and smartphone systems, e.g. viruses, worms, spywares etc. The name is a combination between the two words malicious and software, hence, meaning malicious software [61].
- MITM:** Abbreviation for Man-In-The-Middle and it is a type of attack where the attacker sits between two communicating parts and intercepts (and possibly interacts with) their communication without the two parts noticing it, i.e. they think they talk directly with each other [1].
- MMS:** Abbreviation for Multimedia Messaging Service and it is an extension to SMS (see SMS) so that messages can contain multimedia contents instead of only text [32].
- OS:** Abbreviation for Operating System.
- PGP:** Abbreviation for Pretty Good Privacy and it is a software that uses encryption to establish secure communications over internet and for digital signing. PGP can for instance be used for encrypting and decrypting emails, texts and files [59].
- PIN:** Abbreviation for Personal Identification Number.
- PIN message:** PIN message is a BlackBerry feature that makes it possible to send small messages

11 Definitions and abbreviations

	between BlackBerry devices by addressing the receiver with the device's unique id-number (PIN) [23].
PKCS#5:	Abbreviation for Public Key Cryptography Standard number 5 and it is a standard algorithm for generating symmetric encryption keys from passwords. This standard is provided by RSA [59].
Platform:	A platform is the architecture of a system including the underlying hardware, the operating system, the software framework and the runtime environment [62].
PPTP:	Abbreviation for Point-To-Point Tunneling Protocol and it is a network protocol used with VPN to open a remote connection from a client to a host in the network. PPTP is opening up a direct tunnel to a peer inside the network [63].
RSA SecurID:	A SecurID is a token developed by RSA and is used for two-factor authentication for systems. The token generates a one-time password every 60 seconds based on the current time and a token specific secret key. These tokens are used together with passwords to establish two-factor authentication [64].
SHA-1/512:	SHA is abbreviation for Secure Hash Algorithm and is a collection of cryptographic hash-functions. A hash-function takes an input of any size and generates an output of fixed size from the input. It should not be possible to regenerate the input from the output and it should be impossible to determine before the calculation what output a specific input will give. SHA-1 generates a 160-bit output and SHA-512 generates 512-bit output [59].
SIM-card:	SIM is an abbreviation for Subscriber Identity Module and a SIM-card is a type of smart card that is used in mobile devices to identify the subscriber for a mobile phone. The card contains information such as the ICCID, IMSI and more [1].
Smartphone:	Smartphone is the combination between a mobile phone and a computer. Smartphones can be used for phone calls, messaging, web browsing, playing games and much more.
SMS:	Abbreviation for Short Message Service and it is text messages that are sent over the mobile communication system GSM [32].
Spyware:	Spyware is a type of malware that is spying on the user and collecting user information such as social security number, addresses and so on [61].
SSID:	Abbreviation for Service Set Identifier and it is the identification string of a wireless network. This id is used by hosts in a network to establish which network they currently belong to [65].
SSL/TLS:	Abbreviation for Secure Socket Layer/Transport Layer Security and those are security features built upon cryptographic protocols. These features are used for encrypting communication between two communicating parties in a network. Both protocols are using asymmetric encryption to distribute session keys and then a symmetric encryption algorithm is used for encrypting the communication [1].
Trojan:	A trojan is a type of malware where malicious code is embedded in another software or file so that the user does not know that it is malicious. When the software/file is used is the malicious code executed as well [61].
USRP:	Abbreviation for Universal Software Radio Peripheral and it is a hardware device used

for implementing software-based radios.

- Virus:** A virus is a type of malware, which is designed to spread by itself. Viruses are copying themselves into files and softwares in the system and when an infected software/file is being used the malicious code is executed as well [61].
- VPN:** Abbreviation for Virtual Private Network and it is a service that can be used to in a secure way provide access to private networks from a remote distance [63]. This service is often used for accessing corporate networks remotely.
- Wi-Fi:** Is the technique behind the wireless network standard IEEE 802.11b/g/n and it is used for creating wireless local area networks and wirelessly granting devices internet access [65].
- Worm:** A worm is a type of malware, which is designed to be able to spread and execute without the need of user actions. Worms often use known vulnerabilities to be able to spread themselves and execute malicious code on an infected system [61].

12 References

- 1: Gollmann, Dieter (2011). Computer Security (3 ed.). John Wiley & Sons.
- 2: Hogben, Giles & Dekker, Marnix. (2010). Smartphones: Information security risks, oportunities and recommendations for users.
- 3: Apple iOS 4. (2010). <http://www.apple.com/iphone/ios4/>. Accessed 2011-02-04.
- 4: Security Focus column. (2011). <http://www.securityfocus.com/columnists/269>. Accessed 2011-02-04.
- 5: iOS Technology Overview. (2010).
<http://developer.apple.com/library/ios/#documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html>. Accessed 2011-02-08.
- 6: iPhone 4 Teardown. (2010). <http://www.ifixit.com/Teardown/iPhone-4-Teardown/3130/1>. Accessed 2011-02-10.
- 7: Avraham, Itzhak. (2010). Non-Executable Stack ARM Exploitation.
- 8: Security Overview. (2010).
http://developer.apple.com/library/ios/#documentation/Security/Conceptual/Security_Overview/Introduction/Introduction.html. Accessed 2011-02-10.
- 9: Heider, Jens & Boll, Matthias. (2011). Lost iPhone? Lost Passwords!.
- 10: iOS Application Programming Guide. (2010).
<http://developer.apple.com/library/ios/#documentation/iphone/conceptual/iphoneosprogrammingguide/Introduction/Introduction.html>. Accessed 2011-02-10.
- 11: iOS 4: Understanding data protection. (2010). <http://support.apple.com/kb/HT4175>. Accessed 2011-02-10.
- 12: Blazakis, Dionysus. (2011). The Apple Sandbox.
- 13: PaX ASLR. (2008). <http://pax.grsecurity.net/docs/aslr.txt>. Accessed 2011-03-14.
- 14: Find your iPhone or iPad. (2011). <http://www.apple.com/mobileme/features/find-my-iphone.html>. Accessed 2011-02-14.
- 15: iPhone Configuration Utility. (2010).
http://developer.apple.com/library/ios/#featuredarticles/FA_iPhone_Configuration_Utility/Introduction/Introduction.html. Accessed 2011-02-14.
- 16: iPhone Enterprise. (2011). <http://www.apple.com/support/iphone/enterprise/>. Accessed 2011-02-14.
- 17: App Store Resource Center. (2011). <https://developer.apple.com/appstore/resources/submission/>. Accessed 2011-02-14.
- 18: X-Gold 618 Product Brief. (2008).
<http://www.intel.com/products/wireless/mobilecommunications/baseband/wcdma-hspa.htm>. Accessed 2011-02-14.

12 References

- 19: Jurick, David & Stolarz, Adam & Stolarz, Damien (2009). iPhone Hacks (1 ed.). O'Reilly Media.
- 20: BlackBerry 101. (2007). <http://crackberry.com/blackberry-101-lecture-2-bes-and-bis-whats-difference>. Accessed 2011-02-17.
- 21: Security Technical Overview - BlackBerry Internet Service v3.2. (2010). <http://docs.blackberry.com/en/admin/subcategories/?userType=2&category=BlackBerry+Internet+Service+security>. Accessed 2011-02-17.
- 22: Security Technical Overview - BlackBerry Enterprise Server v5.0.2. (2010). <http://docs.blackberry.com/en/admin/subcategories/?userType=2&category=BlackBerry+Enterprise+Server+Security>. Accessed 2011-02-17.
- 23: Introduction to Platform. (2008). <http://programming4.us/mobile/2538.aspx>. Accessed 2011-02-21.
- 24: BlackBerry Smartphones. (2011). <http://us.blackberry.com/smartphones/>. Accessed 2011-02-21.
- 25: The Case for Elliptic Curve Cryptography. (2009). http://www.nsa.gov/business/programs/elliptic_curve.shtml. Accessed 2011-02-22.
- 26: National Institute of Standards and Technology. (2000). Digital Signature Standard.
- 27: Policy Reference Guide - BlackBerry Enterprise Server v5.0.2. (2010). <http://docs.blackberry.com/en/admin/deliverables/16713/>. Accessed 2011-02-23.
- 28: BlackBerry App World Distribution. (2011). <http://us.blackberry.com/developers/appworld/distribution.jsp>. Accessed 2011-02-24.
- 29: MobiHand. (2011). <http://www.mobihand.com/>. Accessed 2011-02-24.
- 30: BlackBerry Java SDK - Security v6.0. (2011). http://docs.blackberry.com/en/developers/deliverables/21091/Running_BB_Applications_use_protected_API_s_656036_11.jsp. Accessed 2011-02-24.
- 31: Seriot, Nicolas. (2010). iPhone Privacy.
- 32: GSM World. (2011). <http://www.gsmworld.com/>. Accessed 2011-03-28.
- 33: OTA. (2011). <http://www.gemalto.com/techno/ota>. Accessed 2011-05-16.
- 34: Bosen, Bill. (2010). Why Your Cell Phone Can't Keep Secrets.
- 35: Post- och telestyrelsen. (2011). <http://www.pts.se>. Accessed 2011-05-16.
- 36: Ettus Research LLC. (2011). <http://www.ettus.com>. Accessed 2011-05-16.
- 37: Ubuntu. (2011). <http://www.ubuntu.com>. Accessed 2011-05-17.
- 38: GNU Radio. (2011). <http://www.gnuradio.org>. Accessed 2011-05-17.
- 39: OpenBTS. (2011). <http://www.openbts.sourceforge.net>. Accessed 2011-05-17.
- 40: Asterisk PBX. (2011). <http://www.asterisk.org>. Accessed 2011-05-17.

- 41: Wireshark. (2011). <http://www.wireshark.org>. Accessed 2011-05-17.
- 42: Apvrille, Axelle. (2011). OpenBTS for dummies - v0.5.
- 43: Jammer - The price of silence. (2011). <http://www.jammer-store.com>. Accessed 2011-05-17.
- 44: Architectural Risk Analysis. (2005). <https://buildsecurityin.us-cert.gov/bsi/articles/best-practices/architecture/10-BSI.html>. Accessed 2011-05-19.
- 45: Peltier, Thomas R. (2010). Information Security Risk Analysis (3 ed.). Auerbach Publications.
- 46: Jailbreak. (2011). <http://www.jailbreaking.com>. Accessed 2011-05-19.
- 47: Phishing attacks on smartphone users. (2011). <http://www.trusteer.com/blog/mobile-users-three-times-more-vulnerable-phishing-attacks>. Accessed 2011-05-19.
- 48: Kelsey, John & Schneier, Bruce & Wagner, David & Hall, Chris. (1998). Cryptanalytic Attacks on Pseudorandom Number Generators.
- 49: Nohl, Karsten. (2010). Attacking phone privacy.
- 50: Marlinspike, Moxie. (2009). New Techniques for Defeating SSL/TLS.
- 51: BlackBerry falls to WebKit browser attack. (2011). <http://www.zdnet.com/blog/security/pwn2own-2011-blackberry-falls-to-webkit-browser-attack/8401>. Accessed 2011-05-19.
- 52: Nmap Security. (2011). <http://nmap.org>. Accessed 2011-05-20.
- 53: Kryptos. (2011). <http://www.kryptoscommunications.com>. Accessed 2011-05-31.
- 54: Nova Vision. (2011). http://www.novavisioninc.com/pages/prd_tamper_evident_seals.html. Accessed 2011-05-31.
- 55: BlackBerry WebKit Vulnerability. (2011). <http://www.blackberry.com/btsc/search.do?cmd=displayKC&docType=kc&externalId=KB26132>. Accessed 2011-05-31.
- 56: International Telecommunication Union Press Release. (2009). http://www.itu.int/newsroom/press_releases/2009/49.html. Accessed 2011-06-08.
- 57: Truphone. (2011). <http://www.truphone.com>. Accessed 2011-06-16.
- 58: BlackBerry Smart Card Reader. (2010). <http://us.blackberry.com/atagance/security/products/smartcardreader/>. Accessed 2011-06-17.
- 59: Wobst, Reinhard (2007). Cryptology Unlocked (4 ed.). John Wiley & Sons.
- 60: Oracle Java. (2010). <http://www.oracle.com/technetwork/java/>. Accessed 2011-03-28.
- 61: Filiol, Eric (2005). Computer Viruses - From theory to applications (1 ed.). Springer Editions.
- 62: Stallings, William (2009). Computer Organization and Architecture (8 ed.). Pearson Education.
- 63: Forouzan, Behrouz A. (2009). TCP/IP Suite (4 ed.). McGraw-Hill Higher Education.

12 References

64: RSA SecurID. (2011). <http://www.rsa.com>. Accessed 2011-03-28.

65: WiFi Alliance. (2010). <http://www.wi-fi.org/>. Accessed 2011-03-28.

13 Appendixes

13.1 Appendix A: iOS layer and frameworks overview

Application Layer

<No frameworks>

No frameworks are provided by this layer.

Cocoa Touch Layer

Address Book UI framework

Provides interfaces that are used for presenting address book views in applications.

Event Kit UI framework

Provides interfaces that are used for presenting calendar events in applications.

Game Kit framework

Provides the possibility to use peer-to-peer connections in applications.

iAd framework

Provides the possibility to advertise using banners in applications.

Map Kit framework

Provides a map interface that can be used in applications.

Message UI framework

Provides support for creating and sending emails inside the application by using the user's mail account.

UIKit framework

Provides interfaces for creating and handling user interfaces and much more.

Media Layer

Assets Library framework

Provides an interface and support for retrieving media files from an iOS device, i.e. photos and videos.

AV Foundation framework

Provides routines for playing and handling audio on an iOS device.

Core Audio frameworks

Provides audio services. There are actually three audio frameworks in this category: CoreAudio, AudioToolBox and AudioUnit.

Core Graphics framework

Provides interfaces for 2D graphic drawing.

Core MIDI framework

Provides ways to communicate with MIDI devices such as musical keyboards.

Core Text framework

Provides interfaces for handling text layouts and fonts.

Core Video framework

Provides support for buffers and buffer pools for playing media on an iOS device.

Image I/O framework

Provides interfaces for importing and exporting image data to and from an iOS device.

Media Player framework

Provides support for play media in an application.

13 Appendixes

OpenAL framework	Open Audio Library provides standard interfaces for rendering positional audio in applications. Often used in games.
OpenGL ES framework	Provides tools and interfaces for draw 2D and 3D graphics in applications.
Quartz Core framework	Provides interfaces for composing advanced animations and visual effects in applications.

Core Service Layer

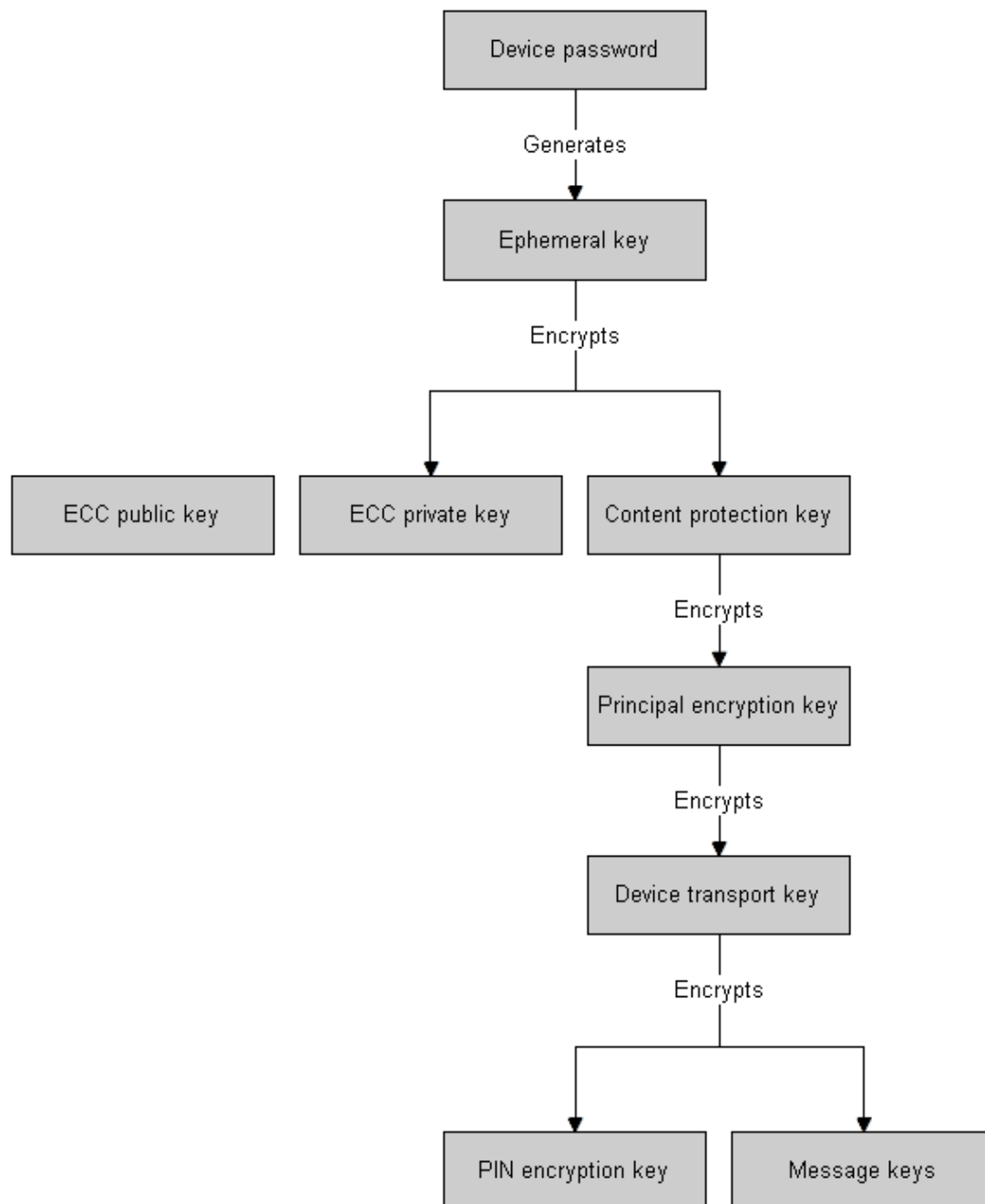
Address Book framework	Provides access to an iOS device address book programmatically so contacts can be displayed and modified in applications.
CFNetwork framework	Provides support for working with network protocols, stacks and sockets. It also provides support for SSL/TLS.
Core Data framework	Provides support for managing data models for applications and makes it easier to define data structure for the developer.
Core Foundation framework	Provides interfaces for data management and features for applications in C, e.g. date and time management.
Core Location framework	Provides locations in form of latitude and longitude coordinates to applications.
Core Media framework	Provides low-level support for media types that can be used on an iOS device.
Core Telephony framework	Provides interfaces for handling phone information and make it available for applications, e.g. phone service provider.
Event Kit framework	Provides access to the user's calendar events on an iOS device programmatically so events can be displayed in applications.
Foundation framework	Provides objective-C wrappers for many of the features in Core Foundation framework described above.
Mobile Core Services framework	Provides low-level definitions for types in Apple's Uniform Type Identifiers.
Quick Look framework	Provides an interface to support previewing of files of types that the application does not has support for opening.
Store Kit framework	Provides support for purchase services within an application, e.g. upgrade from a trial version to a full version of the application.
System Configuration framework	Provides interfaces for accessing information about network connections, preferences and configuration of an iOS device.

Core OS Layer

Accelerate framework	Provides interfaces for performing advanced mathematical computations that are optimized for the hardware of the iOS device.
External Accessory framework	Provides support for accessing and communicating with hardware accessories that are connected to an iOS device in some way.
Security framework	Provides interfaces for handling certificates, encryption keys, policies and pseudorandom numbers on an iOS device.
System	It is the kernel of the OS so it is not an actual framework but it provides some low-level features through the LibSystem library.

[5]

13.2 Appendix B: BlackBerry encryption key hierarchy



[22]

13.3 Appendix C: OpenBTS modification

The modification has to be done in the file `<OpenBTS ROOT>/Control/MobilityManagement.cpp`.

The source code must be inserted in the function `LocationUpdatingController(...)` at line 239, after the code line:

```
else LOG(INFO) << "registration ALLOWED: " << mobID;
```

The source code to add:

```
//PATCH FOR AUTO-UPDATE FOR ASTERISK WHEN NEW IMSI FOUND

//Create a file pointer
FILE *p_file;

//Open file sip.conf in append mode
p_file = fopen("/etc/asterisk/sip.conf", "a");

//Create a new SIP user named IMSI.... with correct context in sip.conf
fprintf(p_file, "\n[IMSI%s]\ncanreinvite=no\ntype=friend\nallow=gsm\ncontext=sip-local\nhost=dynamic\n", mobID.digits());

//Close file sip.conf
fclose(p_file);

//Open file extensions.conf in append mode
p_file = fopen("/etc/asterisk/extensions.conf", "a");

//Generate an internal extension number for this IMSI (Number between 2000-2999)
int extension = (rand() % 1000 + 2000);

//Write extension number and call macro action to extensions.conf
fprintf(p_file, "\nexten => %i,1,Macro(dialGSM,IMSI%s)\n",extension, mobID.digits());

//Close file extensions.conf
fclose(p_file);

//Reload asterisk configuration
system("/etc/init.d/asterisk reload");
```


13.4 Appendix D: Risk tables

Risk 1:	Risk of disclosure of sensitive and private information to an agent through jailbreaking/rooting attacks			
Vulnerabilities:	Implementation bug that leads to that root access is possible, careless users			
Platform:	iOS		BlackBerry OS	
Usage:	Business	Regular	Business	Regular
Complexity level:	Low (4)	Low (4)	Very high (1)	Very high (1)
Impact level:	Catastrophic (5)	Severe (4)	Catastrophic (5)	Severe (4)
Risk level:	Critical (20)	Important (16)	Low (5)	Negligible (4)

Risk 2:	Risk of disclosure of sensitive and private information to an agent through malware attacks			
Vulnerabilities:	Insufficient malware protection, user unawareness			
Platform:	iOS		BlackBerry OS	
Usage:	Business	Regular	Business	Regular
Complexity level:	High (2)	Medium (3)	Medium (3)	Low (4)
Impact level:	Catastrophic (5)	Severe (4)	Catastrophic (5)	Severe (4)
Risk level:	Intermediate (10)	Intermediate (12)	Important (15)	Important (16)

Risk 3:	Risk of disclosure of sensitive and private information to an attacker through phishing attacks			
Vulnerabilities:	Small devices displays, user unawareness			
Platform:	iOS		BlackBerry OS	
Usage:	Business	Regular	Business	Regular
Complexity level:	Low (4)	Low (4)	Low (4)	Low (4)
Impact level:	Catastrophic (5)	Severe (4)	Catastrophic (5)	Severe (4)
Risk level:	Critical (20)	Important (16)	Critical (20)	Important (16)

Risk 4:	Risk of disclosure of sensitive information to an agent through attacks on device encryption			
Vulnerabilities:	Insecure key generation, weak passwords			
Platform:	iOS		BlackBerry OS	
Usage:	Business	Regular	Business	Regular
Complexity level:	High (2)	Medium (3)	Very high (1)	Very high (1)
Impact level:	Catastrophic (5)	Severe (4)	Catastrophic (5)	Severe (4)
Risk level:	Intermediate (10)	Intermediate (12)	Low (5)	Negligible (4)

Risk 5:	Risk of disclosure of sensitive information to an agent through network attacks			
Vulnerabilities:	Implementation bugs, configuration mistakes			
Platform:	iOS		BlackBerry OS	
Usage:	Business	Regular	Business	Regular
Complexity level:	High (2)	High (2)	High (2)	High (2)
Impact level:	Catastrophic (5)	Severe (4)	Catastrophic (5)	Severe (4)
Risk level:	Intermediate (10)	Low (8)	Intermediate (10)	Low (8)

Risk 6:	Risk of financial loss for the user by an agent stealing the device			
Vulnerabilities:	Device value, device size and mobility, careless users			
Platform:	iOS		BlackBerry OS	
Usage:	Business	Regular	Business	Regular
Complexity level:	Medium (3)	Low (4)	Medium (3)	Low (4)
Impact level:	Significant (3)	Catastrophic (5)	Significant (3)	Catastrophic (5)
Risk level:	Intermediate (9)	Critical (20)	Intermediate (9)	Critical (20)

Risk 7:	Risk of financial loss for the user by malware using expensive calls/SMS or services			
Vulnerabilities:	Insufficient malware protection, user unawareness			
Platform:	iOS		BlackBerry OS	
Usage:	Business	Regular	Business	Regular
Complexity level:	High (2)	Medium (3)	Medium (3)	Low (4)
Impact level:	Significant (3)	Catastrophic (5)	Significant (3)	Catastrophic (5)
Risk level:	Low (6)	Important (15)	Intermediate (9)	Critical (20)

Risk 8:	Risk of financial loss for the user by an attacker that access the device over internet to make calls/send SMS			
Vulnerabilities:	Implementation bugs, configuration mistakes			
Platform:	iOS		BlackBerry OS	
Usage:	Business	Regular	Business	Regular
Complexity level:	High (2)	High (2)	High (2)	High (2)
Impact level:	Significant (3)	Catastrophic (5)	Significant (3)	Catastrophic (5)
Risk level:	Low (6)	Intermediate (10)	Low (6)	Intermediate (10)

Risk 9:	Risk of that the user is being surveilled by an attacker using spyware			
Vulnerabilities:	Insufficient malware protection, user unawareness			
Platform:	iOS		BlackBerry OS	
Usage:	Business	Regular	Business	Regular
Complexity level:	High (2)	Medium (3)	Medium (3)	Low (4)
Impact level:	Catastrophic (5)	Severe (4)	Catastrophic (5)	Severe (4)
Risk level:	Intermediate (10)	Intermediate (12)	Important (15)	Important (16)

Risk 10:	Risk of that the user is being surveilled by an attacker eavesdropping on the mobile network data			
Vulnerabilities:	GSM usage and GSM flaws, trust in network			
Platform:	iOS		BlackBerry OS	
Usage:	Business	Regular	Business	Regular
Complexity level:	Medium (3)	Medium (3)	Medium (3)	Medium (3)
Impact level:	Catastrophic (5)	Severe (4)	Catastrophic (5)	Severe (4)
Risk level:	Important (15)	Intermediate (12)	Important (15)	Intermediate (12)

Risk 11:	Risk of that the user is being surveilled by an attacker through a tampered device			
Vulnerabilities:	Careless users, exchangeable parts			
Platform:	iOS		BlackBerry OS	
Usage:	Business	Regular	Business	Regular
Complexity level:	Low (4)	Low (4)	Very low (5)	Very low (5)
Impact level:	Catastrophic (5)	Severe (4)	Catastrophic (5)	Severe (4)
Risk level:	Critical (20)	Important (16)	Critical (25)	Critical (20)

Risk 12:	Risk of that an attacker makes a device or device service unavailable by using malware			
Vulnerabilities:	Insufficient malware protection, user unawareness			
Platform:	iOS		BlackBerry OS	
Usage:	Business	Regular	Business	Regular
Complexity level:	High (2)	Medium (3)	Medium (3)	Low (4)
Impact level:	Significant (3)	Minor (2)	Significant (3)	Minor (2)
Risk level:	Low (6)	Low (6)	Intermediate (9)	Low (8)

Risk 13:	Risk of that an attacker makes a calls and message unavailable by using mobile networks attack			
Vulnerabilities:	Sensitive signals, GSM usage and GSM flaws			
Platform:	iOS		BlackBerry OS	
Usage:	Business	Regular	Business	Regular
Complexity level:	Medium (3)	Medium (3)	Medium (3)	Medium (3)
Impact level:	Significant (3)	Minor (2)	Significant (3)	Minor (2)
Risk level:	Intermediate (9)	Low (6)	Intermediate (9)	Low (6)

Risk 14:	Risk of that an attacker makes a device or device service unavailable by hardware tampering			
Vulnerabilities:	Careless users, exchangeable parts			
Platform:	iOS		BlackBerry OS	
Usage:	Business	Regular	Business	Regular
Complexity level:	Low (4)	Low (4)	Very low (5)	Very low (5)
Impact level:	Significant (3)	Minor (2)	Significant (3)	Minor (2)
Risk level:	Intermediate (12)	Low (8)	Important (15)	Intermediate (10)

Risk 15:	Risk of that user credentials is leaked to an attacker through malware			
Vulnerabilities:	Insufficient malware protection, user unawareness			
Platform:	iOS		BlackBerry OS	
Usage:	Business	Regular	Business	Regular
Complexity level:	High (2)	Medium (3)	Medium (3)	Low (4)
Impact level:	Catastrophic (5)	Catastrophic (5)	Catastrophic (5)	Catastrophic (5)
Risk level:	Intermediate (10)	Important (15)	Important (15)	Critical (20)

Risk 16:	Risk of that user credentials is leaked to an attacker through phishing attacks			
Vulnerabilities:	Small devices displays, user unawareness			
Platform:	iOS		BlackBerry OS	
Usage:	Business	Regular	Business	Regular
Complexity level:	Low (4)	Low (4)	Low (4)	Low (4)
Impact level:	Catastrophic (5)	Catastrophic (5)	Catastrophic (5)	Catastrophic (5)
Risk level:	Critical (20)	Critical (20)	Critical (20)	Critical (20)

Risk 17:	Risk of that user credentials is leaked to an attacker through spoofed Wi-Fi access points			
Vulnerabilities:	Accepts insecure networks, user unawareness			
Platform:	iOS		BlackBerry OS	
Usage:	Business	Regular	Business	Regular
Complexity level:	Low (4)	Very low (5)	Low (4)	Very low (5)
Impact level:	Catastrophic (5)	Catastrophic (5)	Catastrophic (5)	Catastrophic (5)
Risk level:	Critical (20)	Critical (25)	Critical (20)	Critical (25)

Risk 18:	Risk of that user credentials is leaked to an attacker through network attacks			
Vulnerabilities:	Implementation bugs, configuration mistakes			
Platform:	iOS		BlackBerry OS	
Usage:	Business	Regular	Business	Regular
Complexity level:	High (2)	High (2)	High (2)	High (2)
Impact level:	Catastrophic (5)	Catastrophic (5)	Catastrophic (5)	Catastrophic (5)
Risk level:	Intermediate (10)	Intermediate (10)	Intermediate (10)	Intermediate (10)

Risk 19:	Risk of that the user is being surveilled by an attacker through eavesdropping on data traffic by using spoofed Wi-Fi access points			
Vulnerabilities:	Accepts insecure networks, user unawareness			
Platform:	iOS		BlackBerry OS	
Usage:	Business	Regular	Business	Regular
Complexity level:	Low (4)	Very low (5)	Low (4)	Very low (5)
Impact level:	Catastrophic (5)	Catastrophic (4)	Catastrophic (5)	Catastrophic (4)
Risk level:	Critical (20)	Critical (20)	Critical (20)	Critical (20)

Copyright

The publishers will keep this document online on the Internet – or its possible replacement – from the date of publication barring exceptional circumstances.

The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: <http://www.ep.liu.se/>.