Security and trust for mobile phones based on virtualization

Chrystel Gaber, Jean-Claude Paillès

Redaktør: Patrick Bours, Norwegian Information Security Laboratory (NISlab),
Gjøvik University College

Norwegian Information Security Conference
Norsk Informasjonssikkerhetskonferanse

NISK 2010

Gjøvik University College, Gjøvik
23.–24. november 2010

Program Chair

Patrick Bours	HiG

Program Committee

Kristian Gjøsteen	NTNU	Vladimir Oleshchuk	UiA
Tor Helleseth	UiB	Anders Paulshus	Conax
Erik Hjelmås	HiG	Ragnar Soleng	UiT
Audun Jøsang	UNIK	Nils Kalstad Svendsen	HiG
Martin Gilje Jaatun	SINTEF IKT	Svein Willassen	Svein Willassen AS
Stig F. Mjølnes	NTNU	Eli Winjum	FFI
Leif Nilsen	UNIK	Andre Årnes	HiG

Norwegian Information Security Conference
Norsk Informasjonssikkerhetskonferanse

NISK 2010

Gjøvik University College, Gjøvik
23-24 November 2010

Program Chair

Patrick Bours	HiG

Program Committee

Kristian Gjøsteen	NTNU
Tor Helleseth	UiB
Erik Hjelmås	HiG
Audun Jøsang	UNIK
Martin Gilje Jaatun	SINTEF IKT
Stig F. Mjølnes	NTNU
Leif Nilsen	UNIK
Vladimir Oleshchuk	UiA
Anders Paulshus	Conax
Ragnar Soleng	UiT
Nils Kalstad Svendsen	HiG
Svein Willassen	Svein Willassen AS
Eli Winjum	FFI
Andre Årnes	HiG

ISBN 978-82-519-2705-5

Redaktør: Patrick Bours, Norwegian Information Security Laboratory (NISlab), Gjøvik University College

Tapir Akademisk Forlag har som målsetting å bidra til å utvikle gode læremidler og alle typer faglitteratur. Vi representerer et bredt fagspekter, og vi gir ut ca. 100 nye titler i året. Vi samarbeider med forfattere og fagmiljøer i hele landet, og våre viktigste produktområder er:

Læremidler for høyere utdanning
Fagbøker for profesjonsmarkedet
Vitenskapelig publisering

Forlagsredaktør for denne utgivelsen:
Lasse.Postmyr@tapirforlag.no

Tapir Akademisk Forlag
7005 TRONDHEIM
Tlf.: 73 59 32 10
Faks: 73 59 32 04
E-post: post@tapirforlag.no
www.tapirforlag.no

Preface

Welcome to NISK 2010, the third edition of the Norwegian Information Security Conference. After the initial NISK conference in Agder and its follow up in Trondheim, it will now take place in Gjøvik on the 23^{rd} and 24^{th} of November. As before the conference will take place in combination with NIK and NOKOBIT. NISK2010 is sponsored by NISnet, the resource network of Norwegian Information Security researchers funded by the Norwegian Research Council.

This year we had 27 high quality submissions from 8 different institutes. Of those one was withdrawn and one came in too late. The remaining 25 were reviewed by 2 members of the Program Committee each and from their feedback 14 papers were selected for presentation. This means that the acceptance rate of 56% is very close the the 58% from last year. All 14 papers will get a 30 minutes timeslot for presenting the ideas. Out of the 14 papers, 8 are authored or co-authored by PhD students and 1 is co-authored by master students.

We are glad to announce that Dr. Mike Bond from the Computer Laboratory at the University of Cambridge accepted the invitation as a keynote speaker. The title of his presentation is *Chip and Empiricism: Breaking EMV, with proof*. In May 2010 Mike Bond presented the controversial paper *Chip and PIN is broken*, which he co-authored with Steven J. Murdoch, Saar Drimer, and Ross Anderson, at USENIX Security. The paper described how an EMV card can be used to make purchases at Point-of-Sale without knowing the correct PIN. During the subsequent publicity, demonstrations of the technique deployed against the live banking system aired on various European television channels.

I would like to thank all the members of the Program Committee for their valuable input in the reviewing process. Furthermore I would like to thank the organizers of NIK, Erik Hjelmås and of NOKOBIT, Tom Røise for the pleasant cooperation and last but certainly not least I would like to thank Kari Lauritzen for all the help with the practical organization of the three conferences.

Table of Content

NISK 2010

Security and trust for mobile phones based on virtualization

Chrystel Gaber and Jean-Claude Paillès
GREYC laboratory
ENSICAEN - CNRS - University of Caen–Basse-Normandie
14000 Caen, France
cgaber@greyc.ensicaen.fr, jc.pailles@voila.fr

Abstract

In this paper, we present a concept of a trusted computing platform aimed for mobile devices. Most of previous works on trusted computing platforms were aimed for computers. The proposed method is based on the existence of a secure element in the mobile, mobile OS virtualization and trusted boot process. Such a platform is feasible and easy to implement if specific requirements are respected.

1 Introduction

During the past years, the smartphone market has grown quite healthily. The first quarter of 2010 confirms this trend with an increase of 48.7% from the same period in 2009 [1]. Many people have adopted this new type of device for communication and computing. Their interest has grown beyond just managing a calendar and contacts. Smartphones are now expected to be more useful in daily life. They are used to play games, access to the internet, access to e-mails, take pictures or read and modify document.

As electronic transactions seem to become more and more popular, the demand for smartphone applications enabling to purchase goods, to manage one's bank account or other types of transactions will inevitably grow. However, security requirements for these type of applications are quite high and complicated to implement. Moreover, they require limitations to the user-experience which make these measures quite unpopular among users. Lack of freedom and control on devices is one of the reasons why users try to jailbreak or root their phones.

The solution we describe in this paper enables to strengthen the security provided by mobile Operating Systems (OS), to facilitate the implementation of security measures without undermining user experience.

This paper breaks down into five parts. The first part presents the security requirements needed in operating systems to run sensitive applications as well as the current

This paper was presented at the NISK-2010 conference.

features proposed by the most widespread OSs. In the second part, we explain what is our solution to build trust and security in mobile phones. The third part gives an overview of the components required to create such a trusted platform for handheld devices. The fourth part presents the advantages of the proposed architecture. Finally, we conclude this paper.

2 Security services and current implementation

In this part, we will describe security requirements for sensitive applications. We will also have a look how these requirements are enforced in the most common mobile OSs.

Requirements

In information assurance models, five security services are required from the system: availability, integrity, confidentiality, authentication and non-repudiation [2]. The two last services are not taken into account in this paper. Indeed, we consider that these aspects are related to the application's usage. Thus, they should be managed at the application's level and not at the OS's level.

In our context, availability means that no application will prevent another one from carrying out its tasks. For example, it shouldn't be possible for an application A to access the memory allocated to an application B and erase the data needed by application B to run. Confidentiality corresponds to the fact that only legitimate parties can read information. Integrity corresponds to the fact that only legitimate parties can modify information.

Now that we have determined what properties are required from mobile phones to protect sensitive applications, we are going to examine in the following section how these properties are implemented in the most commonly available OSs.

Comparison between the implementation of security services in various OSs

Three criteria have been chosen to evaluate the implementation of the security mentioned above:

- Memory management. In order to provide integrity, availability and confidentiality, memory should be protected.
- Cryptography. For confidentiality and integrity, cryptography should be supported on the OS and cryptography tools should be available for third party application developers.
- Permissions. In order to protect some sensitive APIs and functions which require explicit authorization before being used, permissions are necessary.

Through table 1, we can see that security services are not implemented in the same way in the different OSs. These different implementations also have different levels of efficiency. We can see that most OSs have a technique to manage memory. However, according to [3], the iPhone Os presents some lacks in this field. All these OSs provide

cryptography tools to developers but only Windows Mobile OS supports device encryption. Windows Mobile OS has permission management like Symbian and Android but it isn't as fine-grained. As iPhone is a closed platform, the information concerning its permission management policy has not been found.

	Memory management	Cryptography	Persmissions
Symbian	Processes are protected form one another. Each of them is given a private space which cannot be read by other processes [4]	Cryptographic libraries and plug-ins are available [5]	Sensitive actions can be used if the process has an access token called capability. [5]
Windows Mobile OS	The memory management is based on virtual memory [6]	Device encryption is available. A cryptography API is available [6]	Three permissions levels: privileged, normal and blocked [6]
iOS	According to Apple [7], applications are sandboxed. Yet, according to this analysis [3], memory pages are both writable and readable and there are no heap and stack randomization	Encryption is supported and cryptographic APIs are available [7]	.
Android	Processes are protected from each other. Indeed, each of them are run in a different Dalvik Virtual Machine and have a specific User IDentifier [8, 9, 10]	A cryptographic API is available for developers[11]	Permissions are available and users are informed about the permissions at the application's installation [8, 9, 10]

Table 1: Comparison between OSs

We can conclude that these OSs do not offer the same security guarantees. However, although the security level required for sensitive applications such as conducting financial transactions is quite high, these applications still need the services provided by operating systems. Indeed, these applications may need tools to manage peripherals, to use the network to communicate with a distant merchant server for example or to use the user interface to communicate with the user. The concept presented in the next section aims to solve this problem.

3 Our solution to build trust and security

In this part, we will first explain what assumptions were taken for this work and what are virtualization and trusted boot. These elements are the key points of the architecture we propose. Finally, we will present the solution we propose.

Assumptions

Sensitive applications such as payment applications, banking applications, ticketing applications or applications under subscription contain data and operations which need to be protected.

We assume that mobile phones contain a secure element (SE) which provides security and confidentiality support. We also assume that the most valuable assets of the Service Providers (SP) such as cryptographic keys or sensitive operations are stored in the SE. We consider in this work that the SE is trusted so we don't examine attacks on this element. We take into account attacks against the mobile and its OS. Confidential code theft (PIN for example) can be performed in this way. There is also the possibility for a virus to make a payment instead of the user. Another attack that can be carried out without attacking the SE is to misguide the user by not displaying correct information. This corresponds to the "what you see is what you pay" problem. These types of attacks do not exist for traditional smart cards because service provider terminals can generally be trusted.

Virtualization

Virtualization is based on the use of virtual machine monitors, also called hypervisors. Their role is to enable various operating systems to run on the device in the same time, to monitor their execution and manage the necessary resources (peripherals, memory or CPU for example)[12].

As we can see in figure 1, there are two types of virtualization [12]:

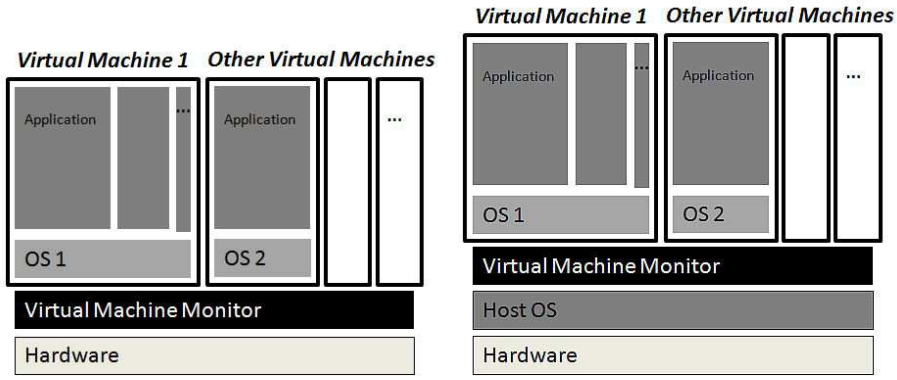
- Type 1 for which the hypervisor runs directly on bare software,
- Type 2 for which the hypervisor runs into a host OS.

Only the first type will be considered in this paper. We consider that for the second type, the hypervisor's security level depends on the host OS whose security level might be difficult to quantify.

The hypervisor's most important contribution for our solution is the complete isolation of the various virtual machines. This will be used to create a secure part in the phone where sensitive applications will run fearlessly.

Focus on trusted boot process

The aim of trusted boot process is to identify whether the platform on which the application is run is sound or not. The platform's status is determined from integrity measures performed on different stages of the boot process. Various definitions of trusted boot such as in the AEGIS platform [13] , the Terra platform [14], and in TCG specifications [15], [16] use the n^{th} level to measure the $(n+1)^{th}$ level's integrity before launching it.



(a) Type 1 (b) Type 2

Figure 1: Types of virtualization

A similar process is used in the architecture we propose. Its purpose is to make sure that the secure part created with virtualization has not been compromised.

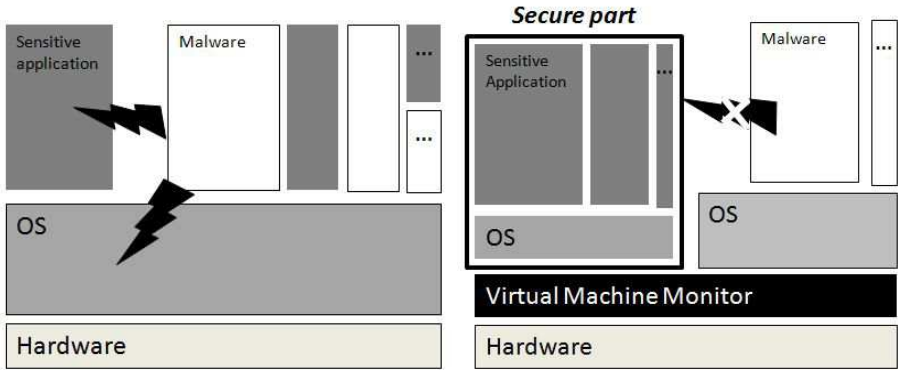
Principle

In section 2, we have seen that sensitive applications need the services of OSs but that native OSs with unknown and unsure applications present many risks. Our solution would be to isolate sensitive applications in a minimalist OS with a strict policy and controlled applications. For this, virtualization is necessary. This idea is close to the Terra platform for trusted computing [14] and is aimed for mobile phones. However, unlike the Terra platform, our approach doesn't oblige application developers to create an Operating System dedicated to their applications.

Figure 2(a) represents the usual mobile architecture whether figure 2(b) represents our architecture's concept. The phone is partitioned into two independent parts. The first one would contain a classical mobile OS. In this part, the user could be entirely free to do whatever he wants and to install whatever application he wants. The second part would be devoted to security-sensitive applications such as payment, electronic signature... It could contain a minimalist OS which could be easy to certify, according to standards such as Common Criteria, and on which a very strict security policy could be enforced.

Virtualization is the means which enables to implement this separation in the phone. Yet, using virtualization does not guarantee that the secure part's trusted OS has not been modified or replaced and hence cannot be relied on any more. The integrity of the mobile's secure part has to be checked in order to avoid this problem. The way to overcome this is to set up a secure boot process.

The proposed boot process, as seen in figure 3, is close to the trusted boot process defined in TCG [15, 16], to the attestation process used on Terra platform [14], and to the AEGIS bootstrap [13]. Our approach is based on two secret values, named R0 and R1. R0 is related to the platform and is stored in a secure zone of the handheld device. We assume that R0 can't be read from outside holds. R1 is a reference value stored in a secure element. It is generally considered that the SIM card is very secure and thus can



(a) Without virtualization (b) With virtualization

Figure 2: Principle of the solution

be considered as the secure element in handheld devices. It corresponds to the value of R at initial state. R is a work register which will contain a measurement of the OS in part 2, the hypervisor and the boot sequence. It is calculated during the boot process and will enable to assess the integrity of the platform. As we can see in figure 3, the boot process takes place according to the following steps:

- the BIOS sets R at R0 (we assume that the BIOS is immutable and may access to R0),
- The BIOS computes M, a hash of the bootloader as well as a hash of R and M. This latest result is stored in R. The BIOS will then launch the bootloader,
- Before loading the hypervisor, the bootloader computes M, which will now be a hash of the hypervisor and update R with the hash of the previous R and M,
- Before loading OS2, the hypervisor would compute the new value of M and R the same way than the previous levels did.

At the end of this boot sequence, R should be equal to R1, if and only if, the boot sequence and some elements in the OS have not been modified. We consider that R is somewhere in memory but can not be read by the external because we assume that the only software attacks that could reveal R's value would need to change the boot sequence and thus would reveal a false value which would not be equal to R1. We also consider that R can't be computed directly from files in permanent memory because we assume that R0 can't be read from outside the device.

The aim of computing R is to perform a handshake between the SIM and the platform and prove to the SIM that the platform is sound. Sending R directly to the SIM would be insecure, because a man in the middle attack is possible between the SIM and the platform. This issue can be solved by using some cryptographic tools. The figure 4 illustrates this sequence. First, the SIM generates a challenge X which is sent to the OS. The OS ciphers the challenge X by using the AES protocol and R as the secret key. The result, Y=AES(R,X) is sent back to the SIM. The SIM then compares Y with Z which is

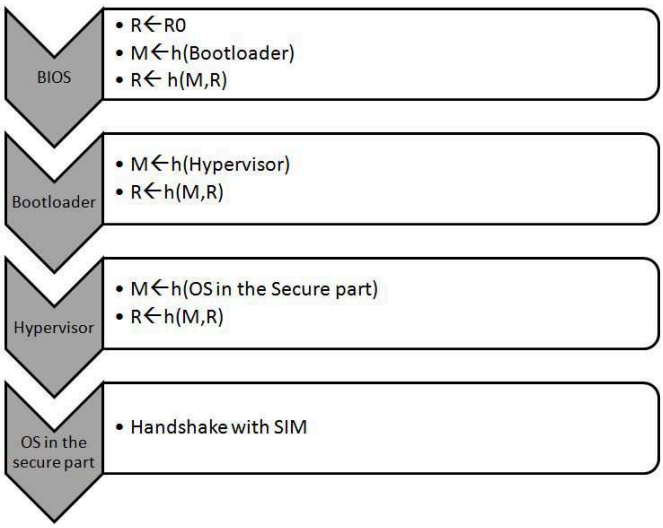


Figure 3: Trusted boot sequence

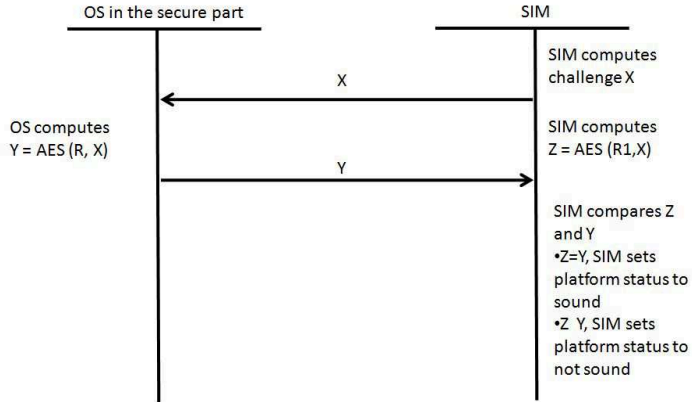


Figure 4: Handshake sequence

calculated by ciphering X with R1 as the secret key of the AES protocol. In this way, it is impossible for a man in the middle to deduce any knowledge on the values R and R1.

We have presented this approach for a SIM, but it can be valid for any other type of secure element.

If this procedure's result indicates that the platform is unsound, the SIM can decide to disable the application's sensitive part which is in the SIM. This approach can be compared to TCG's trusted boot process which influences the key storage functionalities. Table 3 shows the similarities between both approaches.

4 Platform requirements

In this part, we present the requirements related to the different components of the platform. These requirements are necessary to make a viable trusted platform.

Table 2: Comparison between TCG's approach and the proposed one

TCG's approach	Proposed approach
Platform registers and function "extend"	Hash computations and final handshake described above
Predefined values for platform registers	Not covered here
Attestation function to prove platform soundness	SIM is able to contain sensitive data and programs securely and isolated from each others
Platform's data protection	SIM is able to contain sensitive data and programs securely and isolated
Monotonic counters	SIM is able to contain securely and isolated sensitive data and programs, and permanent memory exist
Direct Anonymous Attestation	SIM is able to contain sensitive data and programs, with advanced cryptography, securely and isolated
Multi responsibility	Global Platform is another way to cover this issue. All new SIMs will contain GP features.

Requirements at the platform's initial state

For the solution proposed to be viable, it is necessary to well define its initial state. It is necessary to be sure that the bootloader, the hypervisor and the secure part's OS are in a trusted state. If this is not done, R1 will correspond to an unsafe state of the device and the trusted boot will be null and void.

In order to achieve this, it would be necessary to certify the code of the bootloader, the hypervisor and the OS by a third party.

Requirements related to the hypervisor

Telephony and texting services are the most important services for a handheld devices. They should always be available, whether the user is using the secure virtual machine or the insecure one. Thus, the hypervisor should enable the user to receive phone calls and text messages on whatever part of the phone.

Mobile devices also have specific limitations such as battery management and ressources. These limitations should also be taken into account in the hypervisor.

Requirements related to the Operating System for the secure part

In order to protect the sensitive applications in the secure part, it is necessary for the OS used here to have a very strict security policy. In this OS, there should be a proper isolation between applications. Sensitive APIs or functions should be protected. Moreover, only sensitive and approved applications should be able to be installed in this part. Therefore, certificate management should be supported by the OS and the code of the applications to be installed in this part should be analyzed ans certified by a known third party.

As seen above, this OS should be certified by a third party. It would be interesting to have a minimalist OS to simplify and shorten the certification process.

This operating system should, like the hypervisor, be coded in a way that takes into account the specific limitations of handheld devices. This is also true for the OS of the insecure part.

Requirements for hardware support

As mentioned in [14], a secure User Interface is needed in this type of system. Indeed, we could imagine an attack using an interface that would simulate the interface of the secure part and would register the data entered by the user such as passwords for example. To counter this type of attacks, additional hardware is necessary. We could for example have a small light switching on when the user is on the secure OS. Other OSs should not be able to access this lamp.

As seen in section 3, a trusted zone on the platform would also be useful to store the value R0 which is the base of the trusted boot process. This zone could, for example, be the trusted zone that exists in processors ARM 1176 [17]. This type of processor is adapted for smartphones.

Requirements for the storage of R1 in the Secure Element

In mobile phones, the SE is generally the SIM card. Therefore, if the user of a trusted device wishes to change his mobile to have a new trusted device without changing his SIM card, there could be problems. Indeed, the value R1 stored in the SIM card would not correspond to the integrity measurement of the new platform. Telephony functions would still be possible because the user would still be able to access the insecure OS. However, sensitive applications would be blocked. To solve this problem, there should be a procedure which would enable the SE issuer to update R1 in the SE.

5 Advantages

This method presents some benefits. It is able to satisfy in the same time the user's need of freedom and some Service Providers' need of security. Indeed, the user will be able to do whatever he wants on part one without fearing any consequences on his sensitive applications or data.

This method also facilitates security controls. Indeed, it will be easier and it will need shorter time to certify the hypervisor, the secure partition and its OS, which are small and well controlled, than an entire platform.

Another asset is that this platform doesn't require much additional hardware. A secure zone is needed for R0. This processor is already deployed on some smartphones and can be deployed easily [17]. R1 has to be stored in a secure element too but this secure element corresponds to the SIM card which is already available in the devices we consider. The other type of hardware needed is something to assert that the user is in the secure OS. We can see that most of the hardware necessary already exists. Therefore, this architecture would be fast to be deployed on the market.

6 Conclusion

In this article, we proposed an architecture to create a trusted computing platform for mobiles. Its aim is to enhance security features proposed in mobiles today independently from the OS used. It provides a strong isolation between sensitive applications and normal or malware applications. It also facilitates security controls and certifications needed to assess that a platform is secure for specific sensitive applications. We have also seen that particular requirements are necessary to create such a platform.

The proposed approach presents many advantages. It facilitates the establishment of trust for users and Service Providers without needing new hardware such as a TPM adapted for mobiles. This also implies that it would be easy and fast to be deployed on the market. It also enables to ensure a certain level of security for Service Providers and some freedom to users.

Further works will consist in defining a lifecycle management for R0 and R1 as well as means to update them.

7 Acknowledgements

The material of this contribution results partly of the outcome of a research contract between GREYC Laboratory and Orange Labs.

References

[1] Gartner. Gartner says worldwide mobile phone sales grew 17 per cent in first quarter 2010. http://www.gartner.com/it/page.jsp?id=1372013, May 2010. Last visited on July, 21st 2010.

[2] W.V. Maconachy, C.D. Schou, D. Ragsdale, and D. Welch. A model for information assurance: An integrated approach. In *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security, US Military Academy, West Point, NY*, pages 5–6. Citeseer, 2001.

[3] Charlie Miller, Jake Honoroff, and Joshua Mason. Security evaluation of apple's iphone. Technical report, Independent Security Evaluators, 2007.

[4] Symbian. Memory management concepts. http://developer.symbian.org/main/documentation/reference/s3/pdk/GUID-BFEBCD57-3C83-56D7-B7A3-B8A361725645.html. Last visited on 31st, July 2010.

[5] Symbian. Symbian os v9 security architecture. http://developer.symbian.org/main/documentation/reference/s3/pdk/GUID-1E7AA950-06C2-599C-BCC2-12BB99306E1B.html. Last visited on 31st, July 2010.

[6] MSDN. Security for windows device mobiles. http://msdn.microsoft.com/en-us/library/bb416433.aspx. Last visited on August, 14 2010.

[7] Apple. iphone in businness security overview. Technical report, Apple, 2009.

[8] William Enck, Machigar Ongtang, and Patrick McDaniel. Understanding android security. *IEEE Security & Privacy*, 7(1):50–57, 2009.

[9] Asaf Shabtai, Yuval Fledel, Uri Kanonov, Yudal Elovici, Shlomi Dolev, and Chanan Glezer. Google android: A comprehensive security assessment. *IEEE Security and Privacy*, 8(2):35–44, 2010.

[10] Alan Goode. Managing mobile security: How are we doing? *Network Security*, 2010(2):12–15, 2010.

[11] google. Android package index. `http://developer.android.com/reference/packages.html`. Last visited on August, 1st 2010.

[12] Robert P. Goldberg. *Architectural Principles for Virtual Computer Systems*. PhD thesis, Harvard University, 1973.

[13] William A. Arbaugh, David J. Farber, and Jonathan M. Smith. A secure and reliable bootstrap architecture. In *IEEE Symposium on Security and Privacy*, 1997.

[14] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: A virtual machine-based platform for trusted computing. *ACM SIGOPS Operating Systems Review*, 37(5):206, 2003.

[15] Reiner Sailer, Leendert Van Doorn, and James P. War. The role of the tpm in enterprise security. *Datenschutz und Datensicherheit*, 28:539–544, 2004.

[16] Trusted Computing Group. Trusted computing group specifications. `http://www.trustedcomputinggroup.org/`. Last visited on 6th, August 2010.

[17] ARM. Arm 1176 processor. `http://www.arm.com/products/processors/classic/arm11/arm1176.php`. Last visited on August, 8th 2010.