2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing

Secure mobile business information processing

Nicolai Kuntze, Roland Rieke
Fraunhofer Institute for Secure Information Technology
Darmstadt, Germany
e-mail: {nicolai.kuntze|roland.rieke}@sit.fraunhofer.de

Günther Diederich, Richard Sethmann
Institut für Informatik und Automation
Hochschule Bremen, Germany
e-mail: {Guenther.Diederich|Richard.Sethmann}@hs-bremen.de

Karsten Sohr, Tanveer Mustafa
Technologie-Zentrum Informatik, Universität Bremen, Germany
e-mail: {sohr|tanveer}@tzi.de

Kai-Oliver Detken
DECOIT GmbH, Germany
e-mail: detken@decoit.de

***Abstract*—An ever increasing amount of functionality is incorporated into mobile phones—this trend will continue as new mobile phone platforms are more widely used such as the iPhone or Android. Along with this trend, however, new risks arise, especially for enterprises using mobile phones for security-critical applications such as business intelligence (BI). Although platforms like Android have implemented sophisticated security mechanisms, security holes have been reported [9]. In addition, different stakeholders have access to mobile phones such as different enterprises, service providers, operators, or manufacturers. In order to protect security-critical business applications, a trustworthy mobile phone platform is needed. Starting with typical attack scenarios, we describe a security architecture for Android mobile phones based on the concepts of Trusted Computing. In particular, this architecture allows for a dynamic policy change to reflect the current environment the phone is being used in.**

***Keywords*-Trusted Computing; Android; Mobile Business Processing; Mobile Trusted Module**

I. Introduction

In order to support and improve business decisions and cooperate competitiveness, IT systems are used to collect and process business data. These business intelligence systems (BI systems) strive to combine formerly spread and fragmented data from different parts of a company. Through analysis and transformation, data is turned into information, a basis for strategic decisions. The increased availability and system performance of mobile systems allows flexible on site data collection and processing, thus extending business intelligence to mobile business intelligence. At the same time different wireless communication properties and heterogeneous system environments must be considered. Furthermore, the use of BI tools on mobile devices (e.g., mobile phones, netbooks, smart phones) requires to securely adapt to different physical and networking environments due to a changing work context. The example illustrated in Figure 1 shows different security policies an external service employee is affected by when using a mobile device to securely connect to the BI platform of the customer company.

In the initial state, the employee and his mobile system are part of the service company's network. At this time, the employee has system and data access according to the security policy of his employer's company. Taking up or continuing his work for the customer, the employee switches to a security policy that has been predefined and securely distributed for the purpose of this cooperation. Afterwards he contacts the security gateway of the customer company and authenticates himself. The security gateway verifies the authentication and checks information on the system state of the mobile device (such as the patch status of the client operating system, anti-virus software, firewall, and security settings) against the previously deposited version of the cooperation security policy. If the mobile device lacks mandatory features (e.g., VPN client, up-to-date version of security policy), missing data or software will be provided and the check must be repeated. If the client state complies with the cooperation security policy, a secure VPN connection will be established between the mobile device and a business intelligence portal in the demilitarized zone of the customer network. Using the services of the business intelligence portal, the employee has limited access to the business intelligence platform within the customer intranet. Access is restricted to services needed to fulfill the employee's contract, e.g. getting a service order, accessing data referring to this order, delivering results (new data, reports), closing his own open service orders. Main challenges in this use case include the secure change in security settings and network membership, the reliable collection and secure transmission of the mobile system state as well as the usability and transparency of the corresponding functions.

The contribution of this paper is to describe a solution architecture for the mobile business scenario based on Trusted Computing. This architecture is tailored towards the challenges posed by the nomadic nature of mobile phones. In this sense, these requirements are different from the PC world with a usually fixed environment. To make the discussion more practical, we presume that the mobile BI client runs Android, although our solution is not limited to one specific platform.

The remainder of the paper is organized as follows. Section II briefly describes a security analysis of the mobile BI application in the context of Android. Section III explains the main concepts of Trusted Computing in mobile systems. Section V outlines our security architecture for the mobile BI application based on Trusted Computing considering the

978-0-7695-4322-2/10 $26.00 © 2010 IEEE
DOI 10.1109/EUC.2010.107

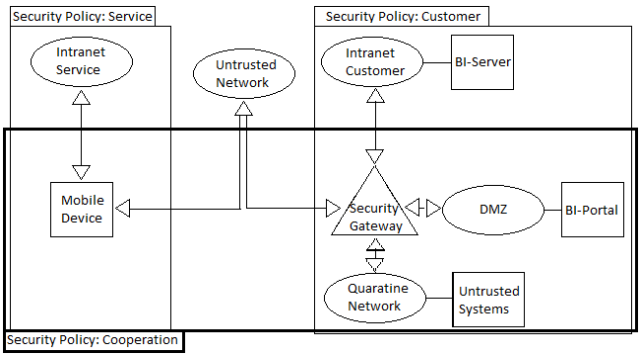

Fig. 1. Different security policies in mobile business intelligence infrastructures

assumptions formulated in Section IV and resulting in conclusions in Section VI.

II. Security analysis

Recall that we are using Android as an illustrative example. We begin by describing several attack vectors possible on that platform. Later in the course of the paper, we demonstrate how our security architecture addresses these security problems. First, we briefly explain the basic Android security concepts.

A. Android security

Android is a Linux platform for mobile phones with a Java middleware on top of the OS. This way, Android applications are usually Java-based, although native code can also be accessed through the Native Development Kit (NDK).

Android has two basic parts of security enforcement [2]. First, applications run as Linux users and thus are separated from each other. This way, a security hole in one application does not affect other applications. However, there is also a concept of inter-process communication (IPC) between different applications, or more precisely, between the Android components of the applications such as activities and services [2]. The Java-based Android middleware implements a reference monitor to mediate access to application components based upon permission labels defined for the component to be accessed. Any application requires an appropriate permission label before it can access a component (mostly, but not necessarily, of another application). A number of features further refine Android's security model. One example is the concept of shared user IDs, i.e., different applications can share the same user ID if they are signed by the same developer certificate. Another refinement are protected APIs: Several security-critical system resources can be accessed directly rather than using components. Examples of such resources are INTERNET (allows an application to open arbitrary network sockets, i.e., to have full access to the Internet) and PROCESS_OUTGOING_CALLS (allows an application to monitor, modify, or abort outgoing calls). In order to mediate access to such resources, additional security checks have been implemented. In addition, permissions are assigned protection levels such as "normal", "dangerous", and "signature".

Permissions are requested during installation and cannot be changed at runtime. The user must decide whether permissions are granted or not and thus decides on the security of the device. Either all requested permissions are granted or the installation will not be completed, which tempts users to grant all requested permissions despite possible security drawbacks.

B. Deducing security requirements from attack scenarios

Let us now assume that the BI application's client runs on the Android platform employing the aforementioned security concepts. Furthermore, we assume that a VPN connection between the mobile client and the BI server exists. Users are authenticated by passwords, and based upon the authentication process, receive roles from an LDAP directory at the server side. A mechanism has been implemented which allows one to change security policies depending on the context the mobile devices are used in (cf. introduction).

An attacker's goal may be to have unauthorized access to company data (industrial espionage) or to manipulate BI data (sabotage). In particular, the following attack scenarios are conceivable on the client side:

- Scenario #1: In the Linux kernel is a security hole (e.g., a buffer overflow) which allows an attacker to become root and for which no patch is available yet (zero day exploit). An end user installs an application exploiting this security hole. As indicated above, it is hard for end users to assess the consequences of confirming security warnings, in particular, if Internet access is requested. The rogue application gathers user credentials (e.g., through key logging) to obtain access to the BI server and then manipulates data in the backend.
- Scenario #2: There is a security bug in the security mechanisms for Android's IPC which allows the attacker to access components of other applications, e.g., through an erroneous implementation of the shared user ID concept [6]. A patch of the Android middleware, however, is available. Further, we assume that the end user has not patched the Android middleware and has installed an application exploiting this hole. An attacker then may have unauthorized access to the mobile BI client and/or to the VPN client.
- Scenario #3: An attacker has reverse engineered the protocol between the mobile client and the BI server and in particular has sniffed credentials of a legitimate user such as passwords. Then the attacker places his own device into the company intranet (e.g., as a service technician) and installs his own "mobile BI client" trying to access the BI server with the help of the credentials.
- Scenario #4: A mobile phone has been stolen or lost. An attacker gets access to the phone and at the same time to the BI application's credentials. This way, he can read from or write data to the BI database.
- Scenario #5: A security policy mechanism for mobile phones has been put in place in the enterprise, i.e., a

phone is only permitted to install an application if the policy allows this. An attacker then might try to attack this policy mechanism. For example, he could try to modify the policy on transition. In addition, he could also attempt to change/erase the policy on the phone.
- Scenario #6: An attacker pretends to be the customer's security gateway, gathering information on the current system status and security policy of the mobile device. In addition to this, the attacker may use the fake security gateway to establish malware or false data on the mobile device.
- Scenario #7: An attacker pretends to be a legitimate mobile client, gathering information on the required measurement values in order to launch a targeted attack on the security gateway of the network.
- Scenario #8: The security functions used by the employees are too complex or their results too intransparent for the employee to verify the security state of his own system. In this case, the employee may be unassured of or might even make wrong assumptions on the system security state.
- Scenario #9: The network with the lower security policy might infect the other network via the mobile device.

At this point, we do not claim that this list is exhaustive. We only want to demonstrate common risks of mobile phone platforms jeopardizing the security of sensitive mobile applications. We devised a set of rules that, when followed by a mobile phone platform, can address the aforementioned attack scenarios:

- Only applications and data that have been authorized by the (service) enterprise's security policy can be installed on the mobile phone. The service enterprise security policy might accept customer company software (e.g. VPN, security policies). The enforcement mechanisms cannot be circumvented by rogue applications. (addresses Scenario #1, #2 and #6)
- The BI server can determine the patch level of the mobile phone. A possibly attacked mobile phone cannot lie w.r.t. this information. (addresses Scenario #2)
- Only devices that have been authorized by the enterprise's security policy can access the BI server. (addresses Scenario #3)
- Compromised or stolen devices are not allowed to access the BI server any more. (addresses Scenario #4)
- Policies must be cryptographically secured in a way that the authenticity and integrity can be guaranteed. Additionally, there is no way to switch off the policy mechanism on the phone. (addresses Scenario #5).
- Systems involved in the business intelligence platform use machine certificates to provide credentials on their identity. (addresses scenario #6 and #7)
- The security functions will be designed for easy use and clear information. Statements on the current security setting are correct, non-ambiguous und correspond to the user level of expertise. (addresses scenario #8)
- The data on the mobile device used in one network must be separated against the use in the other network by role-based access control (RBAC). (addresses scenario #9)

We show in the course of this paper that our security architecture can thwart the aforementioned attack scenarios and in particular satisfies these security requirements. It is noteworthy that the security architecture must guarantee that the enforcement mechanisms for the security policy cannot be circumvented on the phone. In particular, the patch-level status must be communicated trustworthily to the BI server. Android's security mechanisms (e.g., sandboxing applications) do not provide such a level of trust for security-critical applications as vulnerabilities are likely to occur (cf. Scenario #1 and #2).[1] On exploiting security holes in the OS kernel, rootkits could be installed, which, for instance, could lie about the security state of the phone. A rootkit could also overwrite a locally stored policy if no protected memory area (beyond the means of the compromised OS) is provided. Software-based solutions for patching, as commonly used in the PC world, do not help in this scenario since a compromised device can always lie about its current patch level. In order to meet the aforementioned security challenges, a trust anchor on the phone is required which cannot be subverted by a Trojan horse. Providing an architecture based on this trust anchor is the topic of the following sections.

III. Trusted Computing in mobile systems

As shown in Section II, hardware-rooted trust anchors are required to meet the security challenges inherent to the use case in Section I. For the mobile domain, several approaches are already available providing for strong security means [8]. Specifically, ARM trust zones [11] and TI's OMAP approach [3] are potential platforms. Moreover, the Trusted Computing Group [10] introduced concepts for mobile phones allowing for an authentication of behaviour of devices attached to a network. In the following, we present a short overview of Trusted Computing and the implications for mobile devices.

A. Trusted Computing

The Trusted Computing technology as defined by the Trusted Computing Group [5] is a technology that implements consistently behaving computer systems. This consistent behavior is enforced by providing methods for reliably checking a system's integrity and identifying anomalous and/or unwanted characteristics. These methods depict a trusted system's base of trust and thus are implemented in hardware, as it is less susceptible to attacks than the software counterparts.

To successfully realize stringently reliable modules, several cryptographic mechanisms are implemented as a tamper-proof hardware component, namely the Trusted Platform Module (TPM). This chip incorporates strong asymmetric key cryptography, cryptographic hash functions and a random number generator that is capable of producing true random numbers

[1] For a more detailed security analysis of Android see [9]. Specifically, the authors discuss the complexity and modifications of the Linux kernel, which likely make the Android platform error-prone.

instead of pseudo random ones. Additionally, each trusted system is equipped with a unique key pair whose private key is securely and irrevocably stored inside the chip. The chip itself is the only entity to read and use this key for e.g. signing or encryption.

This concept provides a foundation for approving and establishing system integrity because the TPM itself cannot be compromised or spoofed by third party software running on the main processor. This is commonly used to measure system integrity and to ensure a system is and remains in a predictable and trustworthy state that produces only accurate results.

a) Trust for Measurement: The key concept of Trusted Computing is the establishment and extension of trust from an initially trusted security anchor up to other components of a system during boot-up. Each component loaded while booting up the system is measured before execution by computing a SHA-1 digest value of it. The first component of this cycle acts as a security anchor and must be initially trusted since its integrity cannot be measured. This anchor is called *Core Root of Trust for Measurement* (CRTM) and is implemented as an BIOS extension to be executed before any other BIOS code [7]. Thus, the CRTM can measure the BIOS and the platform's firmware. Each subsequent component involved in the boot-up process thereupon measures its successive component. Each measurement is stored in form of hash-chains in *Platform Configuration Registers* (PCRs) on the TPM. These 160-bit registers are in the volatile storage on the chip and can exclusively be updated by calling the TPM command `TPM_EXTEND`. This command includes the old value of a register in the calculation of its new value according to equation 1 thus preventing manipulation of registers.

$$PCR_i = \text{SHA-1}(PCR_i \mid newvalue) \tag{1}$$

This basically implements a non-commutative one-way function preventing from deleting and/or overwriting digest values in a PCR. This also enables tracking of the chronological order in which values were applied to the register. These hash chains stored in PCRs allow for reports on the development of the system since the start of the CRTM. Each PCR is initialized with zeros upon system start and then extended with measured data. Thus, other entities can analyze the current state of a remote system and the history since the last system start. This type of boot-up is called *Trusted Boot Process*. Moreover, the successive process of extending trust with each measurement is commonly referenced to build up a *Chain of Trust*. To reproduce and verify a platform register's value in hindsight, every `TPM_EXTEND` command executed must be tracked in a log. In the case of runtime measurement, this must be done by the operating system resulting in a log called *Stored Measurement Log* (SML).

b) Trust for Reporting: Another main concept of Trusted Computing is Remote Attestation, a process to prove the trustworthiness of a Trusted Platform to an external party. To verify a platform's integrity, a subset of PCRs together with a log of all measurements since startup (*Stored Measurement Log*, SML) is sent to the external party signed by the TPM with a so-called Attestation Identity Key (AIK). The PCR values can then be compared with re-calculated values using the chronological order of measured components logged in the SML. Measurements include all events related to the start of software during the boot phase of a system and later on as part of the operation of the running system. From the SML, no insight on the performed actions of loaded applications can be gained as it only documents that a certain software was started. AIKs represent pseudonymous identities. So-called privacy CAs certify that a particular AIK was generated in a TPM with a particular *Endorsement Key* (EK), the *Root of Trust for Reporting* (RTR). The privacy CA also checks platform and EK certificates. The EK could identify a particular TPM and is therefore (for privacy reasons) not used for signing.

B. Mobile requirements and concepts

As discussed in the use case presented in Section I, mobile environments provide for specific requirements towards devices used and their infrastructure. Specifically, ownership and exposure to hardware based attacks are different to the standard PC use case where the equipment is under the direct (physical) administrative control of the owner. Due to the high risk of theft and malicious users protection of mobile devices must also consider direct (physical) attack vectors.

Due to the operation of the mobile devices in hostile environments means are required to ensure the correct state of the device. These means must take into account the operation of the device in a potentially hostile environment. As shown in Section II, one important aspect is to ensure that data stored on the mobile device is only accessible if the device is in an appropriate state. Therefore, a device internal reference and verifier is required to enforce this particular state e.g. using Reference Integrity Measurement certificates (see [1]). Enforcement is implemented using a secure bootstrap process. Such a secure boot does not only allow for status reporting, but also verifies the measurement value before starting the respective component.

Multiple stakeholders being active on one device are a second constraint specific to the mobile domain. A single mobile device is operated by different stakeholders such as the network operator, service provider, or the end user. Isolation of these different interests is required as it is shown for example by Kasper et al. [4].

The TCG [10] provides for an early standard introducing concepts for mobile phones based on a so-called Mobile Trusted Module. Isolation and state enforcement are core concepts in this specification.

IV. Platform assumption

In accordance with the requirement specifications of mobile devices and the BI scenarios, we specified the following platform assumptions for the first prototype. The core element of the platform is represented by the VPN gateway. Additionally, a management server (e.g., RADIUS), a directory

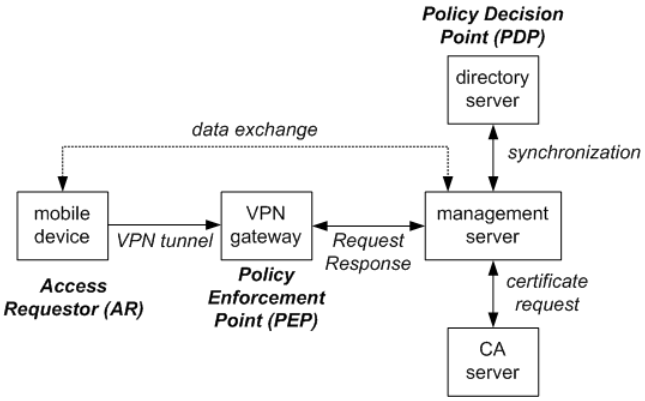

Fig. 2. Platform overview

server (e.g., LDAP), and a certification authority server are necessary. In the first step, the user must be identified for the correct access with the VPN gateway. All criteria are available on the directory server and assign the user to different profiles and user groups. Each user group has different security policies for different access rights. The management system synchronizes the user information with the directory server in regular intervals. This implies that users from the directory server with VPN access rights, even if they are not yet available on the management server, will synchronize with all user group membership automatically after some time. As an option, a public certification authority (CA) can be adapted. If a new user is created on the management server, a certificate will apply. The management server platform is then a registration authority. The VPN gateway must be configured in a way that all requested clients will be authenticated via the management server. Therefore, the gateway site does not need adaptations for a new user. That will be done automatically by the communication with the management server.

Next to the authentication of the user, our platform checks the hardware of the mobile device using the specification Trusted Network Connect (TNC) from the Trusted Computing Group (TCG) with the Mobile Trusted Module (MTM). After the establishment of a VPN connection, the network access of the mobile device is limited to the quarantine zone. Within this area, it is only possible to update software components of the mobile device like anti-virus-software or operating system patches. Access to other network areas of an enterprise network is not permitted. Information about the status of the mobile device is available by the access requestor (AR) on the client-site. The AR includes the network requestor (as a component of the VPN client), the TNC client (as an interface between the network access requestor and plug-in software), and the integrity measurement collector (describes the plug-ins which allow different software products like anti-virus software to communicate with TNC).

In detail, the following steps initiate a mobile device communication as depicted in Figure 2:

1) A VPN connection is established.
2) The management server (TNC server) initializes an integrity check.
3) The mobile device (TNC client) collects integrity measurement (IM) information using the local Integrity Measurement Clients (IMC) on the mobile device (see also Figure 3).
4) The management server (TNC server) forwards the IM information for a check to the integrity measurement verifier (IMV).
5) The integrity measurement verifier (IMV) checks the IMs and sends the results with a recommendation to the management server (TNC server).
6) The management server (TNC server) takes the access decision und forwards this information to the VPN gateway (PEP) and the mobile device (AR).
7) The VPN gateway (PEP) allows or does not allow the access to the network for the mobile device (AR).

Summarized, the integration of the MTM allows for an in-depth check of the software components on the mobile device. This simplifies the detection of rootkits. Furthermore it is possible to sign and encrypt messages with key material of the MTM. That means a strong security check of the origin of the information.

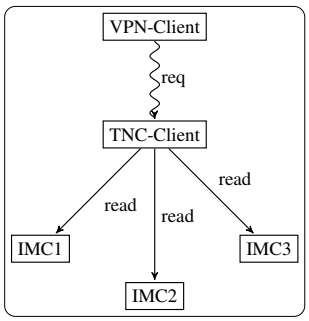

Fig. 3. Functional component model - client

V. Solution architecture

In the classic approach to policy enforcement, Policy Decision Points (PDPs) and Policy Enforcement Points (PEPs) are considered as centralised entities under the assumption that all devices are operated in a controlled environment. In the use case considered, devices are operated in (potentially) hostile environments where even the end users cannot be considered as friendly as devices may get stolen or due to conflicting interests in the business case.

Local enforcement of policies on the device allows for an extended control of the device owner into the leaves of the infrastructure establishing a trust relation into the device. Figure 4 depicts the split of the PDP and PEP functionalities into the domains of the device-side policy enforcement and the infrastructure point of view. Both sets of policies need to be derived from one central policy definition to allow for coherence.

Acceptance of third party policies on the end device needs to be rooted into the policies enforced by the local PDP. Order/priorities of policies are required to ensure that no third party policy conflicts with owner policies and interests. Therefore, we propose (i) to split PDP and PEP into device- and infrastructure-related entities, (ii) to establish policies reflecting the split of concerns, (iii) to assure that PEPs and PDPs on the device and infrastructure are in relation to each other to guarantee proper functionality, and (iv) to support multiple stakeholders by introducing isolation.

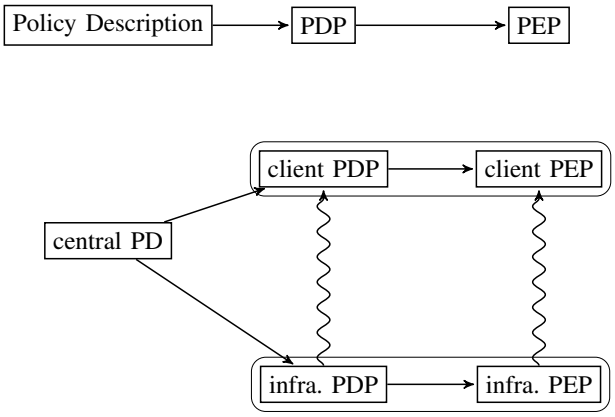

Fig. 4. Decomposition of PEP and PDP

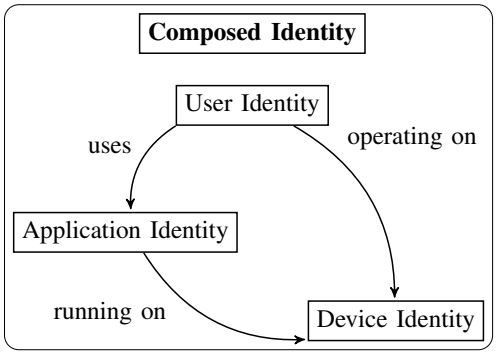

Fig. 5. Definition of Identity for mobile devices

In standard PC architectures, the status of the device used is not considered in the authorisation of actions. Actions are authenticated through the relation to a certain user and his presented identity credentials, respectively. To take the requirements from Section II into account, additional identity definitions for the device and software are required. This extension to the definition of an identity is shown in Figure 5. An identity is composed of the identity of the end user based on well-known identity credentials like passwords, smart cards, or biometric features. Additionally, the device identity and configuration are considered. This identity is based on a unique and unforgeable credential embedded in the device and the state of the device as well as its configuration. The third factor is the status and configuration of the additional software components installed on the device. To allow for an integration of this extended model as presented in Figure 5 with well-known access control models like RBAC, a composed identity is introduced which combines the three presented aspects.

In the mobile use case discussed in Section I, authorisation of actions not only requires the proper identification of the respective user, but also the attestation of the behaviour of the device used by the user. Trusted Computing approaches offer the concepts of remote attestation and, as an extension defined by the Mobile Phone Working Group, an internal verifier as part of the MTM.

Attestation of behaviour performs measurements of the software and its configuration before the execution of the software. These measurements are then verified, either by an internal entity or a remote party. Software used on a mobile device can be differentiated into device functionality provided by the manufacturer and additional software provided and used by the owner.

VI. Conclusion

Using mobile devices in different environments requires a secure transition between different security settings compliant to the corresponding environment. Virtual system environments enable the use of separate system settings, including data separation, on the same mobile device. Trusted Computing mechanisms allow for the integrity of the system state being checked. Transparent use of these key functionalities provides a reliable insight to the security state of the mobile device and therefore the secure transition between different security settings of mobile devices. The proposed architecture demonstrates a possible solution for the compliant use of a mobile device in different security environments, based on a new combination of currently available technologies. Switching security environments begins with launching a virtual environment and contacting the security gateway of the target network. Using Trusted Computing functionality, the security gateway checks the integrity of the system as well as the security settings. On a failed check, the connection will be refused or limited to a quarantine network and the user of the mobile device will be informed. After a successful check, additional setup steps may be carried out, e.g., installing a VPN client and repeating the integrity check, or a secure connection will be allowed to use services in a DMZ. The user of the mobile device may now use these services to work on critical databases like a BI-server in a customer's intranet.

Acknowledgements

This work was supported by the German Federal Ministry of Education and Research (BMBF) under the grant 01IS09032 (VOGUE project).

References

[1] K. Dietrich, M. Pirker, T. Vejda, R. Toegl, T. Winkler, and P. Lipp. A practical approach for establishing trust relationships between remote platforms using trusted computing. *Lecture Notes in Computer Science*, 4912:156, 2008.

[2] W. Enck, M. Ongtang, and P. McDaniel. Understanding Android Security. *IEEE Security and Privacy*, 7(1):50–57, 2009.

[3] P. Gumming. The TI OMAP Platform Approach to SoC. *Winning the SoC revolution: experiences in real design*, page 97, 2003.
[4] M. Kasper, N. Kuntze, and A. U. Schmidt. On the deployment of mobile trusted modules. In *Proceedings of the IEEE Wireless Communications and Networking Conference WCNC 2008, Las Vegas, USA*, 2008.
[5] C. Mitchell et al. Trusted Computing. *Trusted computing*, page 1, 2005.
[6] Open Source Cert Advisory. #2009-006—Android improper package verification when using shared UIDs, 2009. http://www.ocert.org/advisories/ocert-2009-006.html.
[7] S. Pearson. Trusted computing platforms, the next security solution. *HP Labs*, 2002.
[8] S. Ravi, A. Raghunathan, and S. Chakradhar. Tamper resistance mechanisms for secure embedded systems. In *Proceedings of the 17th International Conference on VLSI Design*, page 605. IEEE Computer Society Washington, DC, USA, 2004.
[9] A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, S. Dolev, and C. Glezer. Google Android: A comprehensive security assessment. *IEEE Security and Privacy*, 99(PrePrints), 2010.
[10] Trusted Computing Group. TPM Specification Version 1.2 Revision 103. *Trusted Computing Group*, 2009.
[11] J. Winter. Trusted computing building blocks for embedded linux-based ARM trustzone platforms. In *Proceedings of the 3rd ACM workshop on Scalable trusted computing*, pages 21–30. ACM New York, NY, USA, 2008.