Modeling Trusted Computing Support in a Protection Profile for High Assurance Security Kernels

Hans Löhr[1], Ahmad-Reza Sadeghi[1], Christian Stüble[2], Marion Weber[3], and Marcel Winandy[1]

[1]Horst Görtz Institute for IT-Security, Ruhr-University Bochum, Germany
{hans.loehr, ahmad.sadeghi, marcel.winandy}@trust.rub.de

[2]Sirrix AG, Bochum, Germany
stueble@sirrix.com

[3]Bundesamt für Sicherheit in der Informationstechnik (BSI), Bonn, Germany
marion.weber@bsi.bund.de

Abstract. This paper presents a Common Criteria protection profile for high assurance security kernels (HASK-PP) based on the results and experiences of several (international) projects on design and implementation of trustworthy platforms. Our HASK-PP was motivated by the fact that currently no protection profile is available that appropriately covers trusted computing features such as trusted boot, sealing, and trusted channels (secure channels with inherent attestation).
In particular, we show how trusted computing features are modeled in the HASK protection profile without depending on any concrete implementation for these features. Instead, this is left to the definition of the security targets of a an IT product which claims conformance to the HASK-PP. Our HASK protection profile was evaluated and certified at evaluation assurance level five (EAL5) by the German Federal Office for Information Security (BSI).

1 Introduction

Industrial and governmental IT applications pose a high degree of assurance on the security of the deployed IT products. Consequently, appropriate evaluation means are desired to verify product claims. In this context, *Common Criteria* standards [1] are established methodologies to provide assurance that the process of specification, implementation and evaluation of an IT security product has been conducted in an appropriate, rigorous and standard manner. In particular, protection profiles (PP) define a set of requirements for a specific class of products that must be fulfilled by any product that is certified as compliant to the profile.

For secure operating systems, a small number of protection profiles exist. However, until recently, the existing protection profiles either model only specific aspects such as access control models, or they define the operating system

on a very low level. In particular, these protection profiles do not consider important security aspects that can be realized by the emerging trusted computing technology such as secure booting, trusted channels, or data binding.

For example, the *Trusted Computing Group* (TCG), an industrial initiative aiming at the realization of trusted computing, has specified security extensions for commodity computing platforms. The core TCG specification is the *Trusted Platform Module* (TPM) [2], currently implemented as cost-effective, tamper-evident hardware security module embedded in computer mainboards. It allows a platform to provide evidence of its integrity, cryptographically bind data to previously taken integrity measurements, and protect cryptographic keys in shielded hardware. Based on these functionalities, a secure operating system can realize more advanced protection for applications and more reliable evidence of its trustworthiness to external entities like remote parties.

Using a TPM to realize the mentioned security properties is only one option. Alternative solutions are possible based on other hardware security modules like secure coprocessors [3,4] or smartcards. Hence, to enable the certification of secure systems providing these security properties on an abstraction level allowing end-users to compare security products, a new protection profile incorporating trusted computing becomes necessary.

Contribution. In this paper, we present a Common Criteria protection profile for high assurance security kernels (HASK-PP) [5], based on experience established over several years during the design and development of security kernels in projects such as EMSCB [6], OpenTC [7], and SINA [8]. Moreover, we discuss certain aspects of this protection profile and explain the background of decisions made during the development. The HASK-PP incorporates a number of novelties, compared to existing protection profiles:

- Secure and authenticated boot abstraction (trusted boot)[1]
- User data binding (trusted storage)
- Secure channels with evidence on integrity of endpoints (trusted channels)
- Minimal core security requirements
- High flexibility for implementation

Although one important input to the PP development was trusted computing technology, a strong requirement of the PP development was to keep it implementation-independent. Moreover, a key driver was to minimize the core security requirements, particularly regarding user management and auditing. Only minimal requirements were defined in order to also allow products that do not have (multiple) users or do not need extensive auditing (e.g., embedded devices). The definition of additional security requirements is intentionally left to the specification of security targets of concrete products. All together, this allows a wide range of platforms such as servers, desktop systems, and embedded devices, which can be evaluated according to the HASK protection profile.

[1] We explain the differences between secure and authenticated boot in Section 4.1. In general we use the term *trusted boot* as an abstraction for both.

The protection profile was evaluated and certified at evaluation assurance level five (EAL5) by the German Federal Office for Information Security (BSI).

Outline. In Section 2, we introduce goals and design principles of the development of this protection profile and discuss related and previous work. We also briefly introduce the Common Criteria and relevant terminology. Section 3 presents an overview of the high assurance security kernel protection profile (HASK-PP). We show in Section 4 how trusted computing features are modeled in the protection profile, in particular trusted boot, trusted storage, and trusted channels. Finally, we conclude the paper in Section 5 with a brief summary and an outlook on future work.

2 Toward a Protection Profile for Security Kernels

2.1 Goals and Design Principles

The overall goal of the HASK protection profile was to define evaluation criteria for security kernels that provide functions for the management and separation of compartments operating on top of the security kernel. Examples of product types that may implement these functions are

- Microkernels,
- Virtual machine monitors, and
- Logical partitioning products.

The protection profile was developed based on the experiences with different security kernels covering certain aspects to be considered by HASK:

- *Turaya* [9]: A microkernel-based security kernel for desktop and mobile IT products based on COTS components. An open-source version of the Turaya security kernel has been developed in the EMSCB [6] project partly funded by the German Ministry of Economics and Technology.
- *OpenTC* [7]: A hypervisor-based security kernel for clients and servers, using trusted computing technology. OpenTC is a research project partly funded by the European Union.
- *SINA* [8]: A high-assurance "Secure Inter-Network Architecture" developed by the German Federal Office for Information Security (BSI).

High-level abstraction of trusted computing features such as remote attestation and binding were among the results of these projects. We derived our requirements for a protection profile from these insights. In addition to the security functionality of traditional security kernels (such as access control, audit, etc.), three important functions must exist in a product claiming compliance with the HASK protection profile: (1) trusted channel, (2) trusted storage, and (3) trusted boot.

The first function is the ability to "prove" a "trust status" to a remote trusted IT product and to verify the correctness of a status submitted by a

remote trusted IT product. This status shows that the product is authentic, has not been modified, and is "fresh" (i. e. the status information received has not been replayed from a previous status information potentially intercepted by an attacker). Based on this information, trusted channels between trusted IT products can be established.

The second function allows to bind user data to compartments resp. the security kernel itself (trusted storage). This function can be used to prevent adversaries from bypassing security policies by modification of applications or the operating system. It was deliberately not the goal to prescribe which method is used to implement these functions. However, a product compliant to the protection profile requires hardware, software, or firmware in its environment that is able to ensure the integrity of the security kernel and its data during start-up.

Hence, the third function provides a trustworthy bootstrap mechanism (trusted boot), which supports the other two functions in providing evidence that the product has started in the intended manner. Figure 1 shows the abstract view of a security kernel and our goals.

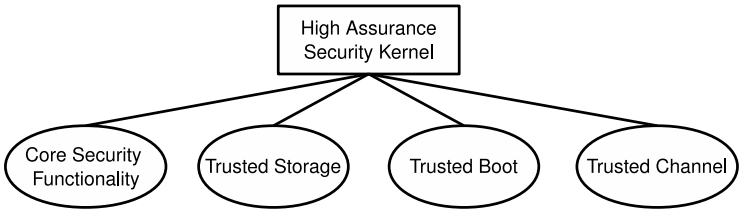

Fig. 1. Abstract functionality of a high-assurance security kernel. "Core Security Functionality" includes separation and access control.

Another important design principle of the HASK-PP was to keep it as minimal as possible to allow a wide range of different realizations, but prevent 'trivial' realizations that do not provide the intended security property from being able to claiming compliance. In fact, the tightrope walk between minimalism and exclusion of trivial realizations was one of the most challenging tasks during the development of this protection profile.

2.2 Common Criteria Basics and Terminology

The *Common Criteria (CC)* are an international standard that aims at permitting comparability between the results of independent security evaluations [1]. The CC provide requirements for security functionality of IT products and assurance measures for the security evaluation of these products. The *Common Criteria Recognition Agreement (CCRA)* regulates international recognition of certificates, and about two dozen countries – including the USA, Canada, UK, Germany, France, Japan, and many others – are currently members of the CCRA.

During security assessment, a given product, the *target of evaluation (TOE)*, is evaluated according to a set of assurance requirements with respect to a *security target (ST)* that defines the security requirements for this TOE. An *evaluation assurance level (EAL)* is a pre-defined set of assurance requirements. The CC specify seven levels (from EAL1 to EAL7), where levels with higher numbers include all requirements from the preceding levels. All hardware, software, and firmware that is necessary for the security functionality of the TOE is called *TOE security functionality (TSF)*. The security requirements that have to be fulfilled by the TSF are called *security functional requirements (SFRs)*. The CC offers a set of classes of pre-defined SFRs, from which designers of security targets can choose. SFR classes are grouped according to security functionality like, e.g., data protection, security management, identification and authentication, auditing.

A *protection profile (PP)* specifies implementation-independent security requirements for a class of TOEs (whereas an ST is implementation-dependent). An ST for a concrete TOE can claim compliance to a PP; in this case, the compliance to the PP is assessed during security evaluation. Protection profiles are particularly important to compare different IT products, since they specify a minimum set of security requirements that must be fulfilled. Of course, the ST for each product can provide additional security features.

2.3 Related Work

The concept of security kernels was explored some decades ago [10,11,12,13,14]. The basic idea is to implement security-critical functionality, i.e., mediating the access to resources according to a security policy, (i) separated from other functionality, and (ii) in a ideally small kernel which allows for the verification of its correctness. The validation and formal verification of security kernels was analyzed and conducted by several works as well [15,16,17,18].

Separation kernels can be seen as a subclass of security kernels. They have only limited functionality. Typically, a separation kernel divides the system into separated partitions running virtual machines. Several commercial companies develop separation kernels, such as LynuxWorks [19], Green Hills Software [20], and Wind River Systems [21]. There is also prior work in formal specification and verification of separation kernels [22,23].

Recently, a protection profile for separation kernels (SKPP) [24] has been introduced and certified in the US. This protection profile has been designed for *high robustness* environments, i.e., it mainly addresses security evaluations at EAL6 and EAL7. The protection profile itself does not claim conformance to a specific evaluation assurance level, but specifies assurance requirements both from EAL6/EAL7, and explicitly defined requirements. Regarding security functional requirements, on the one hand, the focus of SKPP is restricted to the security functionality of separation kernels. In contrast to this, HASK-PP covers a wider range of security functionality, and in particular includes trusted computing functionality. On the other hand, SKPP includes the hardware in the TOE and specifies very detailed security requirements. In contrast, the focus

of HASK-PP is more restricted in this sense, since it excludes the hardware from the TOE and leaves more flexibility for concrete implementations. For a discussion of SKPP and its development, see e.g., [25,26].

Levin et al. [27] compare security kernel and separation kernel architectures with regard to multi-level security. Moreover, they introduce least-privilege separation kernel as a third class of architecture, which supports the security requirements of the SKPP.

In the past, conventional operating systems like various Linux distributions, Unix variants, and versions of Microsoft Windows have been evaluated according to the the *Controlled Access Protection Profile (CAPP)* [28], the *Labeled Security Protection Profile (LSPP)* [29] and the *Role-Based Access Control Protection Profile (RBAC-PP)* [30]. However, these protection profiles target lower evaluation levels[2] and only address the limited aspect of access control models.

3 Overview of HASK-PP

In this section we first describe the architecture and functionality of the TOE (Section 3.1). Then we present an overview of the main components of the protection profile, namely threats and assumptions (Section 3.2) as well as security objectives and security functional requirements (Section 3.3). Finally, we discuss our decision for the evaluation assurance level (Section 3.4).

3.1 Security Kernel Architecture and Functions

The HASK protection profile specifies the security functional and assurance requirements for a class of security kernels that allow executing multiple separated compartments on a single trusted system. Each compartment can behave like a single platform separated from each other with the TOE enforcing this separation and controlling the communication between compartments as well as with external entities in accordance with a defined policy (an overview is shown in Fig. 2). Note that the notion of compartments in the protection profile is a generic concept. A compartment is not necessarily a virtual machine, it can be any set of processes within a security domain. Any product claiming compliance with this PP must provide the necessary security functionality with a high degree of assurance to its users.

To control the communication of external entities with compartments as well as the communication between compartments, the TSF manages a set of *communication objects* that can be assigned to compartments. Communication objects are (on hypervisor-based security kernels) an abstraction for virtual network connections between virtual machines or external networks, and (on microkernel-based security kernels) an abstraction for interprocess-communication between compartmentalized (groups of) processes. Those communication objects allow

[2] While the PPs themselves are certified according to EAL2 and 3, recent evaluations of operating systems according to these profiles achieved EAL4+. However, they are still far from reaching EAL5.

the TSF to control which external entities and other compartments a compartment can communicate with and how this communication is protected. Protection of communication is defined by security attributes assigned to communication objects. Those attributes can define characteristics of the communication link like the set of external entities one can communicate with using this communication object, the kind of protection for the communicated data requested from the TSF when using the communication object (integrity protection, confidentiality protection, authentication of the communication peer).

In addition to the communication objects, the TOE also manages *storage containers* of persistent or volatile storage. Those may be whole disks, disk partitions, disk sectors, etc. where the technology to implement those containers (magnetic disks, flash disks, memory disks etc.) is not relevant for the protection profile.

The security kernel has the following (abstract) set of functions:

- Management of compartments (creation, deletion, starting, changing attributes)
- Management of objects, which are at least containers and communication objects (creation, deletion, changing attributes, defining and managing access control policies)
- Management of resources, which are at least processor time and memory (assignment to compartments, setting resource limits, controlling resource limits)
- Generation and verification of information that reliably shows the integrity of the security kernel, a compartment managed by the security kernel, or specific data. We call such information *evidence of integrity*.

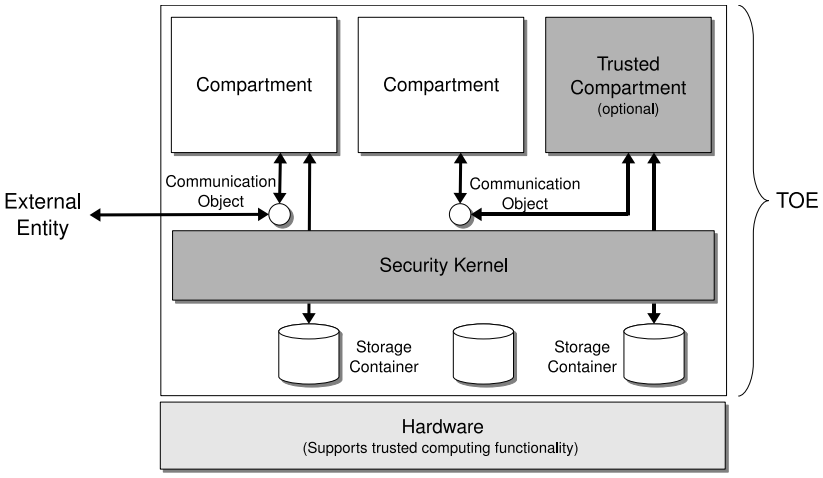

Fig. 2. TOE architecture. Dark gray colored parts implement the TSF.

A security kernel that is compliant to the protection profile needs to implement both mandatory and discretionary access control (MAC/DAC). The DAC policy must at least allow to specify "access" and "no access", and the MAC policy must at least allow separating two compartments from each other such that no information flow between them is possible.

The security kernel (based on hardware functionality required by assumptions in the PP) must be able to protect its integrity, the integrity of compartments, and the integrity of storage containers during runtime. Integrity of the security kernel is obviously required to guarantee the proper operation of the TSF. Integrity of compartments is required for trusted communication channels (see below). Integrity of storage containers is required to prevent unauthorized modification of data when this container is mounted to a compartment.

In a similar way, the system must be able to protect the confidentiality of the security kernel, the confidentiality of compartments, and the confidentiality of storage containers.

Furthermore, the security kernel must be able to provide *trusted channels* between compartments or between compartments and external entities. For a trusted channel, the security kernel has to ensure that the communication link provides integrity and confidentiality protection of the data transferred over the channel, and the identification and authentication of the communication partners must be ensured.

3.2 Threats and Assumptions

The main threats against the TOE include unauthorized access to objects or unauthorized information flow between subjects. Additionally, we considered threats that target to manipulate the TSF or TSF data, including replaying of an older state, e.g., a backup, or influencing the TSF to generate false evidence of the integrity of the TSF or its data. This also includes threats against the TOE environment, e.g., manipulation by installing malicious devices drivers accessing critical hardware functions, or external entities trying to access confidential TSF or user data by starting the TOE outside its intended operational environment.

To address the threats, we stated corresponding security objectives for the TOE and its environment. The latter is important because a security kernel in software alone cannot guarantee or verify its integrity without the assistance of security hardware functionality. In order to be **implementation-independent**, we did not include security functional requirements for the hardware. Instead, we stated **assumptions on the operational environment** in being able to

- support the TOE in producing evidence of the integrity of the TSF code and data during the boot process (`A.INTEGRITY_SUPPORT`);
- allow the TOE to store information such that it cannot be accessed by the TOE where the configuration has been manipulated in an unauthorized way (`A.BIND`[3]);

[3] In TCG terminology, the TPM sealing function can provide such a feature. Other implementations may be based on, e.g., secure coprocessors [3].

– provide a function the TOE trusts that is able to generate evidence of the integrity of a remote trusted IT product only if it is correct (A.REMOTE_TRUST).

The assumptions A.INTEGRITY_SUPPORT and A.REMOTE_TRUST are needed to show the status of integrity at load-time of the TOE. Combined with the integrity protection features of the TSF during runtime, one can derive the integrity status of the running TOE and compartments executed on the TOE.

Of course, for secure operation of the TOE, we assume the environment to

– provide mechanisms for separation of the TSF and other subjects or functionality (A.SEPARATION_SUPPORT);
– not contain backdoors (A.HW_OK);
– not be able to start the TOE in an insecure way without this being detectable (A.NO_TAMPER); and
– not have subjects allowed to perform administrative functions and misusing their privileges (A.NO_EVIL).

3.3 Security Objectives and Security Functional Requirements

The security objectives address protection of objects on the one hand (access control to user data, information flow control between compartments, secure data exchange, management of security attributes, resource limitation to avoid denial of service), and protection of the TSF itself on the other hand (TSF and TSF data integrity and confidentiality). Moreover, the TOE must be able to audit defined potentially security-critical events.

To address the security objectives of the TOE, we defined security functional requirements, which can be assigned to four groups: (i) core security functionality (realizing access control, security management, audit, etc.), (ii) trusted storage, (iii) trusted boot, and (iv) trusted channels. See Figure 3 for an overview (note that the SFRs in the groups overlap because some SFRs are addressing more than one objective).

The core security functionality can be divided into four subgroups[4]. (1) Access control and information flow control includes SFRs for data protection, user identification and authentication, and consistency of TSF data when shared between TSF and another trusted IT product. (2) Resource limitation: As a minimal requirement we include maximum quotas for memory and processor time in order to avoid excessive resource consumption. (3) Audit defines that the TOE must be *able* to audit the following events at minimum: start/stop of audit functions, modifications of security policy enforced by TOE, rejected attempts to perform management operations, and integrity violations of TSF or user data. We did not want to dictate a long list of audit events because some products may not have such strong audit requirement, but should be covered by the HASK-PP,

[4] We introduce the subgroups only as orientation to reflect the objectives in this paper. The reader can find the exact mapping of SFRs to security objectives in the protection profile.

too. The actual selection of events to be audited is up to the security target of a concrete product (and can have even more audit events if necessary).

Finally, (4) security management consists of management of security attributes for subjects and objects, management of TSF data, and management of security roles. The inclusion of these SFRs in the HASK-PP was originally not the focus (because we wanted to minimize the requirements), but they were a result of dependencies between SFRs. For instance, `FDP_ACF.1` (security attribute based access control) requires `FMT_MSA.3` (static attribute initialization).

We discuss the modeling of the objectives trusted storage, trusted boot, and trusted channel in Section 4.

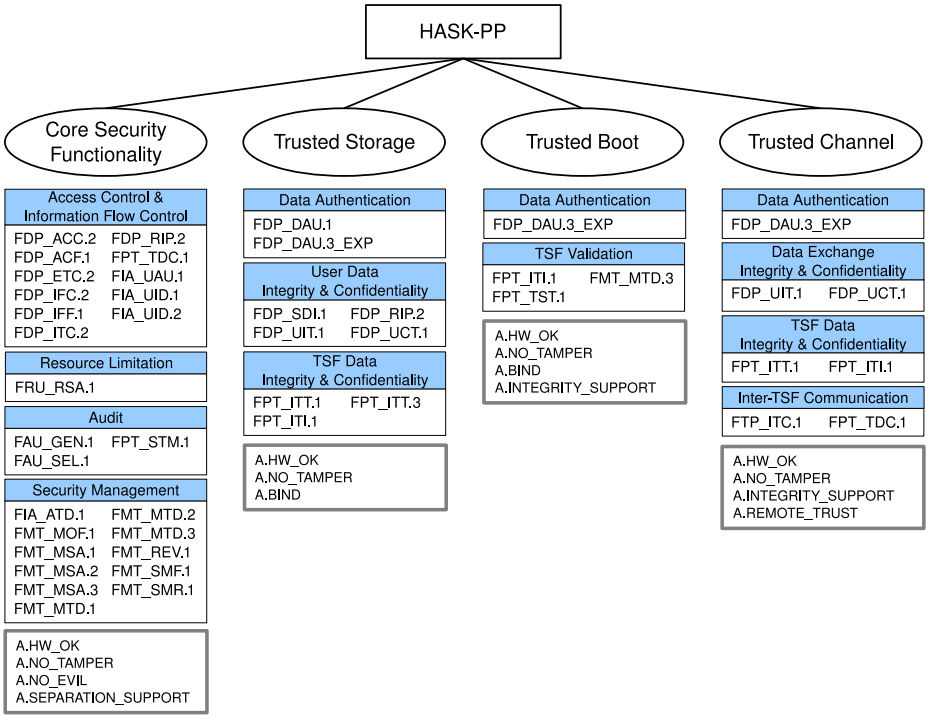

Fig. 3. Overview of the HASK-PP with logical grouping of the security functional requirements and assumptions. See Appendix A for the complete list of SFRs with their names (such as "Basic Data Authentication" for `FDP_DAU.1`).

3.4 Security Assurance

The HASK-PP has been developed against the most recent version 3.1 Revision 2 of the Common Criteria to ensure its usefulness in the future. It is fully

conformant to CC part 3 by selecting the EAL5 package of security assurance requirements.

EAL5 was chosen as a *minimum* level of assurance for different reasons: The architecture of the TOE in HASK-PP addresses systems with exposure to untrustworthy and unauthorized entities and with high value of the data stored and processed by the system. A sufficient level of assurance must be selected to provide system users with appropriate assurance that the system will be able to withstand such threats. The TOEs claiming conformance to the HASK-PP are expected to provide high assurance against the threats assumed in the PP. Robust and reliable separation of compartments requires a level of assurance that includes the evaluation of possible covert channels between unrelated compartments. In this context, testing and vulnerability analysis of the whole TSF is necessary. The whole architecture of a security kernel managing compartments should be implemented in a lean, modularized fashion as required by the EAL5 assurance level. This means to have well-structured internals, a functional specification which is at least semi-formal, and a development process that follows clear implementation standards and defines unambiguous use of development tools.

EAL5 was also deemed appropriate because it shall provide a platform for other secure services implemented in compartments managed by the TOE. Since such services may be certified at assurance levels above EAL4, the underlying platform must not provide weaker assurance.

EAL5 was deemed to be *sufficient* as a minimum level because levels above EAL5 are usually achievable only with extremely high efforts and costs. This allows security kernels to be evaluated and certified according to HASK-PP for commercial application scenarios that do not require the highest levels of assurance. Of course, the security target for any specific product may specify a higher evaluation level. However, the PP itself was only certified at EAL5.

4 Trusted Computing Functionality in HASK-PP

4.1 Modeling Trusted Boot

To be able to provide the "trust status" of the product configuration as required by the HASK-PP, a trustworthy bootstrap architecture is required to convince remote parties about this status. One basic concept towards the development of a trustworthy bootstrap architecture is the so-called chain of trust which has been introduced by Arbaugh et al. [31]. The core idea is that every component involved in the boot process measures the integrity of the succeeding one before it transfers control to it. If the bootstrap process is started by a trusted component (a trusted root host), it is guaranteed that modifications of components involved in the boot process can be detected by the preceding component. Since the relevant product configuration might not be limited to the security kernel itself, we include requirements for loading and starting compartments in the description of the trusted boot process.

In this context, it is possible to distinguish two types of *trusted boot* mechanisms that mainly differ in the way how the measurement results are used:

Definition 1 (Secure Boot). Secure Boot *is a security property of a bootstrap architecture ensuring that only configurations of a certain property can be loaded. If a modification is detected, the bootstrap process is interrupted.*

The term 'property' used in this definition only identifies a set of configurations, e.g., a list or a signature key certifying allowed configurations. An example implementation of secure boot, as proposed by Arbaugh et al., is to verify the integrity of a succeeding component according to a given reference value. If the verification fails, the boot process is halted or an error function is executed [32].

Definition 2 (Authenticated Boot). Authenticated Boot *is a security property of a bootstrap architecture ensuring that remote parties can verify properties of the booted configuration.*

We use the term trusted boot to refer to both secure and authenticated boot. In contrast to secure boot, authenticated boot is not actively influencing the boot process. An example implementation of authenticated boot, e.g., as proposed by the TCG, is to securely store measurement results (i.e., hash values) of the components involved in the boot chain within the Trusted Platform Module (TPM) and attest the values over an authentic channel.[5]

Although both concepts are very similar, they fulfill slightly different security requirements: Secure boot ensures that only valid configurations are loaded. Local users can therefore assume that the platform is in a trustworthy state if the bootstrap process finished successfully. Remote parties, however, can in general not make any assumptions about the loaded platform configuration. In contrast, authenticated boot allows remote parties to verify the platform's configuration. But because any configuration can be loaded, local users can in general not make any assumptions about the current platform configuration. To securely verify the current platform configuration, further mechanisms such as secure hardware tokens or software mechanism are required.

In general, such a trustworthy bootstrap architecture can be realized using different combinations of technologies and assumptions. Typical examples are smartcards, the TPM, a tamper-evident device, or the assumption that adversaries do not have physical access to the platform.

Since such a bootstrap architecture cannot be realized without assumptions regarding the IT-environment (i.e., hardware or environmental assumptions), the HASK-PP models it using the assumptions `A.BIND` and `A.INTEGRITY_SUPPORT`. To allow a compliant product to implement authenticated boot, the assumption `A.INTEGRITY_SUPPORT` only requires that a manipulated security kernel is not

[5] Note that neither authenticated boot, nor secure boot can protect the confidentiality of information under the assumptions that hold for common PC architectures, i.e., the adversary has access to the harddisk. The reason is that both bootstrap architectures do not provide protected storage.

able to generate false evidence of its own integrity. The assumption A.BIND requires that there must be a possibility for the TSF to store data and code in such a way that it can be loaded only if the integrity of the TSF is intact. This allows to implement secure boot.

In the protection profile, several security requirements are related to trusted boot. Existing security functional requirements (SFRs) from the CC have been used to require validation of the security kernel and compartments during start-up. This includes that only secure values for memory and CPU time assigned to a compartment are accepted, that the TSF runs a suite of self tests when loading a compartment that requires integrity evidence, and that any modifications between shut down and start-up of the system (e.g., due to manipulations of the hard disk when the system is shut off) can be detected.

Most notably, one extended SFR, FDP_DAU.3_EXP "controlled data authentication", has been defined specifically for the HASK-PP. This requirement states that the TSF must provide the capability to generate evidence that can be used as a guarantee of the integrity of objects. Moreover, it allows the security target of a concrete product to specify conditions under which such evidence is generated, and subjects must be provided with the ability to verify such evidence. This extended SFR allows the security kernel to extend the "chain of trust" (which must be rooted either in hardware or in the operational environment, as expressed by the assumptions mentioned above) up to individual compartments started by the security kernel. Furthermore, it is also relevant for other trusted computing features, such as trusted storage and trusted channels discussed in the following sections.

HASK-PP requires only that the IT-environment offers a mechanism to check the integrity of the security kernel either before (e.g., by a tamper-resistant cover) or during (e.g., with a TPM) loading it. Which alternative is used is left for the specific implementation to decide.

4.2 Modeling Trusted Storage

A security kernel claiming compliance to the HASK-PP must provide *trusted storage*, according to the following definition:

Definition 3. Trusted storage *is storage where* confidentiality*,* integrity*, and* freshness *(i.e., protection against replay attacks) of stored data is provided, and where the* integrity of the TOE accessing the data *is ensured (in order to prevent other software, such as alternative or modified operating systems, from accessing the data).*

To support trusted storage, a security kernel needs special support from the operational environment, which is reflected in the PP as assumption A.BIND. This assumption requires that the security kernel can store information in such a way that it cannot be accessed by a manipulated TOE (or by software with a different configuration). In concrete systems, A.BIND is usually fulfilled by special hardware features.

Moreover, the security kernel has to ensure that confidentiality and integrity of the data are protected both when the system is running, and when it is offline. Furthermore, it must be infeasible for an attacker to modify the system, or to obtain confidential information from the system in order to get access to the protected data. Additionally, the security kernel must provide a capability to authenticate storage containers, and to verify the integrity of the entity (e.g., compartment) accessing the data. These requirements are expressed by SFRs from the Common Criteria classes FDP (user data protection) and FPT (protection of the TSF), together with the requirement `FDP_DAU.3_EXP` (cf. Section 4.1).

Several possibilities exist to implement trusted storage in real systems. One such possibility is based on a TPM, however, other concepts for trusted computing, e.g., based on proprietary security modules that are not compliant to the TCG specifications, might use different approaches to realize trusted storage.

Trusted storage with the TCG specifications. In the terminology of the TCG, *sealing* denotes the encryption of data with a key that can only be used by a specific TPM under strictly defined conditions: During sealing, the user can specify values for the evidence of integrity that has to be present inside protected registers of the TPM for decrypting (*unsealing*) the data. During unsealing, the TPM checks the content of these registers and refuses to decrypt if the current evidence deviates from the required values. Sealing provides integrity and confidentiality of the data, as well as integrity of the TOE. To support freshness, monotonic counters, another feature of the TPM, can be used.

4.3 Modeling Trusted Channels

The possibility to establish trusted channels has to be provided by any security kernel claiming compliance to HASK-PP.

Definition 4. *A trusted channel is a channel between two entities that provides* integrity, confidentiality, *and* authenticity *of the transmitted data, and ensures* integrity *and* authenticity of the end points.

Hence, a trusted channel allows the communication partners to receive integrity (attestation) information from their peers. A trusted channel may either provide mutual attestation (i.e., integrity measurements of both end points are transmitted), or only the integrity of one end point is verified. Several solutions for trusted channels based on the TCG specifications have been proposed in the literature [33,34,35,36,37].

To keep the protection profile general and implementation-independent, we need to formulate abstract requirements for the trusted channel, without excluding any specific realization.

The hardware and environmental assumptions which are required for a trusted channel are the availability of a mechanism for the TOE to produce evidence of its own integrity (`A.INTEGRITY_SUPPORT`) and the availability of a mechanism (that must be trusted by the TOE) providing a similar feature for the remote entity (`A.REMOTE_TRUST`).

The mandatory functionality of the security kernel to support trusted channels are required by SFRs from the CC for integrity and confidentiality of user data and TSF data during transfer, security functional requirements from the CC for inter-TSF communication, and the component for controlled data authentication that has been introduced specifically for HASK-PP (`FDP_DAU.3_EXP`).

The distinctive feature of *end point integrity* provided by trusted channels is expressed by requiring *assured identification* of the end points in the CC component `FTP_ITC.1` ("Inter-TSF trusted channel"). Here, the term *assured identification* includes integrity verification.

5 Conclusion

In this paper, we describe the first Common Criteria protection profile for a secure operating system with support for enhanced security features, as they are provided by trusted computing technology. The protection profile is general and abstract, thus covering a wide class of IT products without fixing specific mechanisms and leaving a maximum of flexibility and freedom for concrete implementations.

We show how trusted computing features like trusted boot, trusted storage, and trusted channels can be expressed in a generic way by a protection profile, and we point out the relation to existing concepts like the TCG specifications. Moreover, we present and explain the motivation behind the protection profile and important design decisions.

Since the protection profile has been certified, it can be used as a guideline for the design of real systems by security architects. Proof-of-concept implementations and other results from projects like EMSCB, OpenTC, and SINA provide a starting point for developing a security kernel that can be evaluated and certified according to the HASK-PP.

Acknowledgment

This work has been partially funded by the European Commission as part of the OpenTC project [7]. We would like to thank Helmut Kurth and Gerald Krummeck from atsec information security for their invaluable contribution in writing the protection profile, and the anonymous reviewers for their thoughtful comments on this paper.

References

1. Common Criteria for Information Technology Security Evaluation. http://www.commoncriteriaportal.org/thecc.html
2. Trusted Computing Group: TPM Main Specification Version 1.2 rev. 103 (July 2007) https://www.trustedcomputinggroup.org.
3. Smith, S.W., Weingart, S.: Building a high-performance, programmable secure coprocessor. Computer Networks **31**(8) (April 1999) 831–860

4. Yee, B.S.: Using Secure Coprocessors. PhD thesis, School of Computer Science, Carnegie Mellon University (May 1994) CMU-CS-94-149.
5. Kurth, H., Krummeck, G., Stüble, C., Weber, M., Winandy, M.: HASK-PP: Protection profile for a high assurance security kernel. http://www.sirrix.com/media/downloads/54500.pdf (June 2008)
6. European Multilaterally Secure Computing Base. http://www.emscb.de
7. Open Trusted Computing. http://www.opentc.net
8. Sichere Inter-Netzwerk Architektur (SINA). http://www.bsi.bund.de/fachthem/sina/index.htm
9. Sadeghi, A.R., Stüble, C., Pohlmann, N.: European multilateral secure computing base - open trusted computing for you and me. Datenschutz und Datensicherheit DuD **28**(9) (2004) 548–554 Verlag Friedrich Vieweg & Sohn, Wiesbaden.
10. Schroeder, M.D.: Engineering a security kernel for Multics. In: SOSP '75: Proceedings of the fifth ACM symposium on Operating systems principles, New York, NY, USA, ACM (1975) 25–32
11. Walter, K.G., Schaen, S.I., Ogden, W.F., Rounds, W.C., Shumway, D.G., Schaeffer, D.D., Biba, K.J., Bradshaw, F.T., Ames, S.R., Gilligan, J.M.: Structured specification of a security kernel. In: Proceedings of the international conference on Reliable software, New York, NY, USA, ACM (1975) 285–293
12. Chittenden, B., Higgins, P.J.: The security kernel approach to secure operating systems. In: ACM-SE 17: Proceedings of the 17th Annual Southeast Regional Conference, New York, NY, USA, ACM (1979) 136–137
13. S. R., J.A., Gasser, M., Schell, R.R.: Security kernel design and implementation: An introduction. Computer **16**(7) (1983) 14–22
14. Karger, P.A., Zurko, M.E., Bonin, D.W., Mason, A.H., Kahn, C.E.: A retrospective on the VAX VMM security kernel. IEEE Transactions on Software Engineering **17**(11) (November 1991) 1147–1163
15. Kemmerer, R.A.: Formal verification of the UCLA security kernel: abstract model, mapping functions, theorem generation, and proofs. PhD thesis (1979)
16. Millen, J.K.: Security kernel validation in practice. Commun. ACM **19**(5) (1976) 243–250
17. Rushby, J.: Design and verification of secure systems. In: SOSP '81: Proceedings of the 8th ACM Symposium on Operating Systems Principles, ACM (December 1981) 12–21
18. Silverman, J.M.: Reflections on the verification of the security of an operating system kernel. In: SOSP '83: Proceedings of the ninth ACM symposium on Operating systems principles, New York, NY, USA, ACM (1983) 143–154
19. DeLong, R.J.: LynxSecure separation kernel – a high-assurance security RTOS. Technical report, LynuxWorks, San Jose, CA (May 2007)
20. Green Hills Software Inc.: INTEGRITY PC Technology. http://www.ghs.com/products/rtos/integritypc.html (November 2008)
21. Wind River Systems Inc.: Wind River High-Assurance Solutions for Aerospace & Defense. Whitepaper (February 2008) http://www.windriver.com/products/product-overviews/PO_MILS_Solution_Feb2008.pdf.
22. Martin, W.B., White, P.D., Taylor, F.S.: Creating high confidence in a separation kernel. Automated Software Engineering. **9**(3) (2002) 263–284
23. Heitmeyer, C.L., Archer, M., Leonard, E.I., McLean, J.: Formal specification and verification of data separation in a separation kernel for an embedded system. In: CCS '06: Proceedings of the 13th ACM conference on Computer and communications security, New York, NY, USA, ACM (2006) 346–355

24. Information Assurance Directorate: U.S. government protection profile for separation kernels in environments requiring high robustness (SKPP) (2007) available from http://www.niap-ccevs.org/cc-scheme/pp/pp.cfm/id/pp_skpp_hr_v1.03.
25. Nguyen, T., Levin, T., Irvine, C.: High robustness requirements in a common criteria protection profile. In: IEEE International Information Assurance Workshop. (2006)
26. DeLong, R.J., Nguyen, T., Irvine, C., Levin, T.: Toward a medium-robustness separation kernel protection profile. In: ACSAC'07. (2007)
27. Levin, T.E., Irvine, C.E., Weissman, C., Nguyen, T.D.: Analysis of three multilevel security architectures. In: CSAW '07: Proceedings of the 2007 ACM workshop on Computer security architecture, New York, NY, USA, ACM (2007) 37–46
28. National Security Agency: Controlled access protection profile (CAPP) (1999) available from http://www.niap-ccevs.org/cc-scheme/pp/id/PP_OS_CA_V1.d.
29. National Security Agency: Labeled security protection profile (LSPP) (1999) available from http://www.niap-ccevs.org/cc-scheme/pp/id/PP_OS_LS_V1.b.
30. Reynolds, J., Chandramouli, R.: Role-based access control protection profile (RBAC-PP) (1998) CygnaCom Solutions, Inc. and National Institute of Standards and Testing. Available from http://www.niap-ccevs.org/cc-scheme/pp/id/PP_RBAC_V1.0.
31. Arbaugh, W.A., Farber, D.J., Smith, J.M.: A secure and reliable bootstrap architecture. In: Proceedings of the IEEE Symposium on Research in Security and Privacy, Oakland, CA, IEEE Computer Society Press (1997) 65–71
32. Arbaugh, W.A., Keromytis, A.D., Farber, D.J., Smith, J.M.: Automated recovery in a secure bootstrap process. In: Proceedings of the Symposium on Network and Distributed Systems Security (NDSS '98), San Diego, California, Internet Society (2008) 155–167
33. Goldman, K., Perez, R., Sailer, R.: Linking remote attestation to secure tunnel endpoints. In: Proceedings of the 1st ACM Workshop on Scalable Trusted Computing (STC 2006), ACM (2006) 21–24
34. Stumpf, F., Tafreschi, O., Röder, P., Eckert, C.: A robust integrity reporting protocol for remote attestation. In: Proceedings of the Second Workshop on Advances in Trusted Computing (WATC 2006 Fall). (2006)
35. Sadeghi, A.R., Wolf, M., Stüble, C., Asokan, N., Ekberg, J.E.: Enabling fairer digital rights management with trusted computing. In: Proceedings of Information Security, 10th International Conference (ISC 2007). Volume 4779 of Lecture Notes in Computer Science., Springer (2007) 53–70
36. Gasmi, Y., Sadeghi, A.R., Stewin, P., Unger, M., Asokan, N.: Beyond secure channels. In: Proceedings of the 2nd ACM Workshop on Scalable Trusted Computing (STC 2007), ACM (2007) 30–40
37. Armknecht, F., Gasmi, Y., Sadeghi, A.R., Stewin, P., Unger, M., Ramunno, G., Vernizzi, D.: An efficient implementation of trusted channels based on OpenSSL. In: Proceedings of the 3rd ACM Workshop on Scalable Trusted Computing (STC 2008), ACM (2008) 41–50

A Security Functional Requirements

The security functional requirements of the HASK-PP originate all from Common Criteria V3.1 Release 2, part 2, with the exception of FDP_DAU.3_EXP,

which is an extended requirement defined in the protection profile. Table 1 summarizes the security functional requirements of HASK-PP.

Table 1: Security functional requirements in HASK-PP

SFR	Title
FAU_GEN.1	Audit data generation
FAU_SEL.1	Security audit event selection
FDP_ACC.2	Complete access control
FDP_ACF.1	Security attribute based access control
FDP_DAU.1	Basic data authentication
FDP_DAU.3_EXP	Controlled data authentication
FDP_ETC.2	Export of user data with security attributes
FDP_IFC.2	Complete information flow control
FDP_IFF.1	Simple security attributes
FDP_ITC.2	Import of user data with security attributes
FDP_RIP.2	Full residual information protection
FDP_SDI.1	Stored data integrity monitoring
FDP_UCT.1	Basic data exchange confidentiality
FDP_UIT.1	Data exchange integrity
FIA_ATD.1	User attribute definition
FIA_UAU.1	Timing of authentication
FIA_UID.1	Timing of identification
FIA_UID.2	User identification before any action
FMT_MOF.1	Management of security functions behavior
FMT_MSA.1	Management of security attributes
FMT_MSA.2	Secure security attributes
FMT_MSA.3	Static attribute initialization
FMT_MTD.1(1)	Management of TSF data
FMT_MTD.1(2)	Management of TSF data for communication objects
FMT_MTD.2	Management of limits on TSF data
FMT_MTD.3	Secure TSF data
FMT_REV.1	Revocation
FMT_SMF.1	Specification of management functions
FMT_SMR.1	Security roles
FPT_ITI.1	Inter-TSF detection of modification
FPT_ITT.1	Basic internal TSF data transfer
FPT_ITT.3	TSF data integrity monitoring
FPT_STM.1	Reliable time stamps
FPT_TDC.1	Inter-TSF basic TSF data consistency
FPT_TST.1	TSF testing
FRU_RSA.1	Maximum quotas
FTP_ITC.1	Inter-TSF trusted channel