Scaling Concepts between Trust and Enforcement

Andreas U. Schmidt
CREATE-NET Research Centre, Trento, Italy
andreas.schmidt@create-net.org

Andreas Leicher
Johann Wolfgang Goethe-Universität, Frankfurt, Germany
leicher@cs.uni-frankfurt.de

Inhyok Cha
InterDigital Communications, King of Prussia, PA, USA
inhyok.cha@interdigital.com

ABSTRACT
Enforcement and trust are opposite concepts in information security. This chapter reflects on the paradigm shift from traditional concepts of access control and policy enforcement toward decentralised methods for establishing trust between loosely connected entities. By delegating parts of enforcement tasks to trusted elements dispersed in a system, the system can establish transitive trust relationships. This is the most advanced evolution of the organisational method of separation of duties within IT security. The technological basis for trust in systems – Trusted Computing platforms – is described on conceptual levels allowing comparison with other top-level security concepts and mapping to application domains. Important applications in modern information systems and networks are exhibited.

KEYWORDS
IT Security; Trust; Enforcement; Trusted Computing;

INTRODUCTION
One of the major elements of the success of a technology is the adoption and integration by the target group. Therefore, the target group has to put trust, defined as the confidence that a trustor can rely on, in the technology. The adopter can, therefore, allow certain vulnerability on his or her own part. Thus, trust is a process that needs to be understood, and it is an issue that needs to be explored through collaboration from social and technical perspectives as proposed by Cofta (2007).

In IT-Security as an applied science, the last four decades of research have been centred on enforcement as a base concept. On a systemic level, enforcement is the only way to establish security properties with certainty, e.g., carry out formal security proofs, i.e., to exclude every risk. But this also rules out trust and thus has disadvantages in situations where global enforcement is not completely possible. Even if it is possible, it may not be desirable universally due to cost of implementation. On the other hand, trust and its mirror concept of risk inherently include the
notion of the cost, since risks are quantified using expected costs of not meeting them. The growing trend to de-centralised open systems produces numerous situations in which enforcement, by practical necessity, has to be complemented by controlled risk, that is, trust. This chapter emphasises the contradistinction between trust and enforcement with the aim to come to a useful synthesis of both – *scalability of trust in systems*.

In the next part of this chapter, notions of trust are reviewed and integrated into a synthetic definition of trust in technical systems, which can be effective in applications. This is contrasted with traditional notions of enforcement. Then, enforcement and trust technologies are circumscribed systematically on a high level to make the corresponding concepts comparable. Emphasis is on the means to establish trust in systems, since those are relatively young. Some concrete details of the life and operational cycles of trusted systems are given. We then show some recent applications exhibiting how trust and enforcement can go hand-in-hand using decentralisation and separation of duties as core paradigms. Furthermore, future research directions in scalable trust are described. They emerge from the evolution of communication networks and the Internet, where nodes become ever more heterogeneous and connections more ephemeral. Important new insights may also emerge from economics, sociology, psychology, and theories of complex systems that are self-organizing and/or evolutionary.

**BACKGROUND**

According to Dwyer & Cofta (2008), the socio-cognitive model of trust holds that a trustor makes a decision based on an assessment of cues of evidence about a specific situation and a trustee. A more formal definition is given by Gambretta (1988), and Jøsang & Kinateder (2003): “trust (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent will perform a particular action, both before (the trustor) can monitor such action (or independently of his capacity of ever to be able to monitor it) and in a context in which it affects (the trustor's) own action” (p. 213). Trust, as the underlying concept for each economic process needs a good understanding before it can be formalised within a specific model and applied to technology. An important requisite for trust is a risk, or having something invested, as Gambetta (1988) remarks. Castelfranchi and Falcone (1998) extend the definition of Gambretta (1988) to include the notion of *competence* along with *predictability*. In all these definitions, trust is considered as a subjective notion, that is, it is not linked to empirical observation of the trustees’ behaviour. Grandison and Sloman (2000) stress the aspect of *contextuality* of trust, meaning that the expectable actions of the trustor are conditioned by the world state in which they occur. This is also emphasised in the language of information systems by Yahalom, Klein & Beth (1993).

The need to use the concept of trust only arises in a risky situation. The exact relation between trust and risk can be complex (Deutsch, 1958; Mayer, Davis & Schoorman, 1995). So called *trust metrics* (Toone, Gertz & Devanbu, 2003; Kamvar, Schlosser & Garcia-Molina, 2003) are only an intermediate step on the way of constructing this relationship, since they do not yield statistical statements on observable system behaviour. Trust in technical systems and trust building mechanisms between them have long been studied and are varied (Aberer & Despotovic, 2001; Blaze, Feigenbaum & Lacey, 1996).

As we see, there are various meanings to trust between entities, but only a few can be applied without much distortion to the relations between technical systems. In synthesis of the above, we propose to apply the following consistent *operational interpretation of trust* to the relations and interactions between technical systems and between technical systems and human beings:
“An entity can be trusted if it predictably and observably behaves in the expected manner for the intended purpose”

This is essentially also a synthesis of the meanings that TCG and ISO attribute to trust, cf. Pearson (2002b). The operational interpretation, which is actually rooted in physicists’ prevalent understanding of quantum systems (Haag 1992; Lamb, 1969 & 2001), has three salient features:

- **Predictability** designates a priori knowledge about a system that can be used to a) assess the risk incurred in interacting with that system, and b) allow obtaining knowledge about the system during the interaction by reasoning on observations.
- **Observability** specifies the means by, and extent to which knowledge about a system can be gained in interactions. It is closely linked to predictability, in that observations, together with predictions, yield further knowledge on a system’s state and future behaviour.
- **Contextuality** designates information delineating the scope of interactions with the system in which predictions hold and observations can be made.

Formally, all three kinds of information can be expressed in terms of logical predicates and clauses. But their interpretation is essentially *statistical*. This provides the link to the interpretation of trust as effectively mitigated risk. The three properties allow, at least in principle, a mapping between the socio-economic concept of trust and technical concepts. Taken together, they allow an assessment of the trustworthiness of it, or reciprocally, the risk it poses to an interacting entity.

The analogy to physics may seem diagonal, but is conceptually fruitful. For instance, the relation of statistical interpretation to economic risk entails the question of an analogous interpretation of the system-apparatus interaction in measurements. It follows that the notion of contextuality, when applied to the risk incurred, needs to take into account both the cost of failure of the system to behave properly and the cost of observing that. That is, risk and trust will need to rely on a comprehensive cost-benefit analysis including the measures for establishment of trust. We conclude that if the efforts to produce and verify the evidence that can attest to the system’s trustworthiness are too great, then even if it is trustworthy in terms of the bare risk incurred in operating the system, it would be impossible for an observer to actually know that. Therefore, the system cannot be trusted operationally. The reader might feel the resemblance to Heisenberg’s uncertainty relation. It can also be mentioned that, like experimental settings and apparatus’ change, evaluations of operational trust are bound to change as open systems and means to monitor them evolve.

In contradistinction to the observational perspective embodied in our definition of trust, information security takes an active stance with regard to the achievement of *protection goals* (classically confidentiality, integrity and availability of data) within a global information system. It rules out threats by trying to absolutely assure specific system behaviour to a relying party, that is, by enforcing it. The main results of the enforcement approach are twofold. In the interaction between entities, it led to the development of specific *protocols* that systems have to follow to provably reach the desired protection goals. Prime examples are non-repudiation and fair exchange. Enforcement of security by protocols has also shown some principal roadblocks, for instance the impossibility for two parties to perform a fair exchange of an information item without a trusted third party (TTP), as proven by Pagnia, Vogt & Gärtner (2003). The second aspect is the enforcement of system behaviour by *policies*. A policy, according to Dulay, Lupu, Sloman & Damianou (2002) is a “rule governing the choices in behaviour of a managed system”
It requires monitoring the system’s dynamics and continuous matching to the rules set forth in the policies. According to application context, policies come in many flavours and have given rise to many ramifications of research directions, most notably

- **Access Control** focusing on policies for controlling access to documents and resources, in particular authorisation constraints, contextual constraints, and delegation rules.
- **Policy-Based Management** that gives declarative definitions of rules constraining system behaviour. The main advantage is dynamisation of policies, i.e., the possibility to manage them during system operation.
- **Privacy** focuses on the handling of information relating to individual persons such as restricting the communication of, and access to, personally identifiable information based on, e.g., Data Protection regulations.
- **Enterprise Rights Management (ERM)** focuses on the distribution of sensitive information within and between cooperating organisations.
- **Digital Rights Management (DRM)** focuses on the policies applying to the distribution of copyrighted material and in particular media, or in general, digital goods (Becker, Buhse, Günnewig, & Rump, 2003; Schmidt 2008). DRM and ERM are subsumed under the term **Information Rights Management (IRM)**.

It is obvious that in application to real-world systems – which are the empirical basis of computer science, both enforcement and trust stand on feet of clay. In the lack of methods to establish operational trust in distributed systems (Blaze, Feigenbaum & Lacey, 1996) refer to trust as a systematic framework for security provided by specific network services: “It is our thesis that a coherent intellectual framework is needed for the study of security policies, security credentials, and trust relationships. We refer collectively to these components of network services as the trust management problem” (p. 164). Such an approach, without operational trust in the components of a distributed system, does not allow solid assessment of the risk mitigation qualities of security measures therein, let alone a quantitative one.

We conclude that there is a conceptual gap between trust and enforcement, caused by the lack of means to establish operational trust. Such has become more obvious with growing heterogeneity of interconnected systems beyond client-server relationships. In such environments, and given the state-of-the-art of (security) technology, neither enforcement nor the operational view of trust can be realised. Systems lack a) ubiquitous technical means to establish operational trust, b) overarching infrastructures for enforcement, and c) means to convey information on trustworthiness, and applicable security levels to external entities. Only these basic building blocks can enable a dynamical balancing of trust and enforcement reflecting real-world requirements, that is, **scalable trust in systems**.

### ENFORCEMENT TECHNOLOGY

Information systems and technology for the enforcement of policies have come a long way from simple models for control of access to data (La Padula & Bell, 1973; Bell 2005) to means to implement and manage complex, and natural-language security requirements in distributed systems. This also goes beyond the well-known security paradigms in client-server relationships. The main thrust in applied research is the implementation of policies on inter-organisational data exchange using formal methods and IRM. Figure 1 shows a very simplified architecture for this.
In the generation of policies enforceable by an IRM system, three levels of formality can be distinguished: i) a **business level policy** – a human-readable representation of the policy which addresses the risks and threats related to exposing data outside the organisation; ii) a **formal high-level policy** – a representation of the agreement using a formal language, which is suitable for logical analysis and reasoning about the agreement; and ii) an **enforceable (or operational) policy** – a representation of the policy in an executable policy language, which enforces data access and usage according to the policy clauses.

A **Policy Authoring** tool is a (graphical) tool to support the drafting of the business level policy, and to transform the business level representation into the formal one. It rests fundamentally on an **ontology** for the domain that is covered by use cases and business level policies. One of the most popular ontology editors is the open source system Protégé (2009) of Stanford University. It supports, in particular, Web ontology languages such as OWL (2004).

There are two recent, relevant approaches for the translation of business-level policies into formal languages. The project COSoDIS (2009), proposes to write contracts directly in a formal specification language called CL. The modelling of natural English-based policy clauses of a contract into CL is done based on *patterns* (Gamma, Helm, Johnson & Vlissides, 1995; Workflow Patterns, 2007). Then, a translation function is defined that maps the contract into a variant of the *mu-calculus* (Emerson, 1996), after which formal reasoning can be applied by way of the NuSMV model checking tool (Cimatti *et al.*, 2002). The second approach is based on the authorisation language SecPAL of Becker, Fournet & Gordon (2007), in which a grammar of the controlled language of policy clauses is created and then translated into the formal language using parsing and compiling tools.

As policy authoring is mainly a human effort, only supported by tools for partial automation, the outcome is prone to inconsistencies and gaps. The high-level formal policies are, therefore, inspected in a process called **policy analysis** to ascertain that they satisfy the desired properties, in particular protection goals on data and resources. Such is commonly done by model checking tools working on state machines and theorem provers. This area of research is vast; we name only the Athena security protocol analysis tool of Song, Berezin & Perrig (2001) and (Zhang, Ryan, & Guelev, 2005) as non-representative examples. Finally, a complete set of high-level policies is
translated into an enforceable language in an automated process called refinement (Gorrieri, Rensink & Zamboni, 2001; Bandara, Lupu, Moffett & Russo, 2004).

Operational policies are distributed to the systems which are consuming data or accessing resources. The common enforcement architecture deployed on such systems generically consists of a Policy Decision Point (PDP) and a Policy Enforcement Point (PEP). The main reference for this model is the COPS architecture (Boyle, Cohen, Herzog, Rajan & Sastry, 2000; cf. also Law & Saxena, 2003). On a request from an application to access a resource which underlies policy control, the PDP evaluates the policy conditions against locally available data. As a relatively recent development, PDPs may exchange, retrieve, or negotiate policies (Ajayi, Sinnott & Stell, 2008). This may happen on a client-server or peer-to-peer basis.

The decision is passed to the PEP, along with obligations, i.e., those conditions or actions that must be fulfilled by either the users or the system after the decision (Bettini, Jajodia, Wang & Wijesekera, 2002). The PEP exerts control over the resource, for instance by performing an authorized retrieval from a protected external provider, and/or by releasing encryption secrets to the consuming application. The PEP also controls the fulfilment of obligations.

To evaluate policies, the PDP needs information about requesting subject, i.e., the context in which the access request is made and other (internal and external) parameters that are taken into account. The Policy Information Point (PIP) component performs this function. A real-world example of particular importance is the location of a mobile device.

**TRUST TECHNOLOGY**

A system needs certain security-relevant elements and capabilities so that it can be operationally trusted (Rand Corporation, 1970). The ideas to endow systems with trust are not new, and emerged in the context of military applications (Department of Defense, 1985). This context can be viewed as paradigmatic for operational trust – they need to securely operate in situations where any kind of external enforcement may fail and a fallback to inherently trustworthy functionality is a core requirement. Accordingly, the US Department of Defense (1985) differentiates trusted systems into systems and parts of a system of various security levels.

The building blocks of a trusted system establish its trust boundary, and sometimes provide methods to extend it, and to convey trust to an outside entity by making its behaviour and operation predictable and observable to a certain extent. The key techniques in this section comprise (hardware) security anchors, Roots of Trust, Trusted (Sub-) systems and ownership, secure storage and paths, authorisation, authenticated and secure boot processes, and attestation. By combination of these methods, systems can be constructed which combine characteristics of trust and enforcement in manifold ways, and thus enable a scaling of technology between these two poles. In this part of the chapter, we describe the basic functional building blocks.

**Trusted Systems**

A **hardware security anchor** is the key to the protection of the system behaviour. This is a part of the system which is protected against unauthorized access by hardware measures known to be secure enough for the intended purpose to effectively mitigate risks of attacks against it. It holds, in particular, the **Root of Trust (RoT)** for its secure operation. The RoT is an abstract system element which enables

- Securing the internal system operation, and
b) Exposing properties and/or the identity (individually or as a member of a group such as make and model) of the system to external entities in a secure and authentic way.

Genuinely, a system can contain more than one RoT for distinct purposes. Some of them are introduced below. Typical examples for RoTs are asymmetric key pairs together with digital certificates of a trusted third party for them. Also, the symmetric secrets of Subscriber Identification Module (SIM) cards in cellular networks may be viewed as RoTs for the closed, trusted system embodied by the SIM card.

Secondly, functional building blocks in a system that are assumed to be trusted, i.e., to behave in a well-defined manner for the intended purpose, form the Trusted Computing Base (TCB) of the system. The TCB comprises such components of a system which cannot be examined for their operational trust properties when the system is deployed in the field and during operation, but only by out-of-band processes like compliance and conformance testing, and certification. This kind of certification is usually carried out by an independent evaluator, for instance on behalf of the manufacturer of a certain technical element of the TCB or the TCB as a whole, according to established security evaluation standards such as Common Criteria (2009). For such a certification to be useful, the TCB, respectively, its elements need to be endowed with information identifying them as such certified pieces of technology.

A system equipped with defined security anchor, RoTs, and TCB is called a Trusted System (TS). This is a slight refinement of the common notion of Trusted Platforms which, according to Pearson (2002a), is “a computing platform which has a trusted component, probably in the form of built-in hardware which it uses to create a foundation of trust for software processes”, cf. Mitchell (2005). When one or more trusted systems reside within a TS, they are called Trusted Subsystems (TSS). Examples comprise virtual execution environments on a Personal Computer platform which inherit a certain trustworthiness from the hardware Trusted Platform Module (TPM, TCG 2007a) of the host. Another example is the specification of a trusted engine, together with its TCB, in the TCG Mobile Phone Working Group (MPWG) specifications (TCG 2008a, b). In the following, ‘TS’ is interchangeably used as a shorthand for ‘TS or TSS’ where not explicitly stated otherwise.

Below, various capabilities, processes, and architectural elements, summarised under the term trusted resources, TRs, of TS are described. Two kinds of TRs must be generally distinguished: First, TRs which belong to the TCB, and second, TRs which are outside the TCB. Genuine examples for the latter are trusted parts of the operating system, and trusted applications which build on the TCB by using its capabilities. While assertions about the trustworthiness of the TR in the TCB depend on the defined security of the TCB, the trustworthiness of the other TRs can, at most, be derived from that of the TCB. In such a case, the TCB must provide certain internal TRs that allow extension of the trust boundary, i.e., the totality of components of a TS that are considered trustworthy in a given context, to the TRs outside the TCB, for instance authenticated or secure boot as described below. TRs within the TCB often share the same hardware protection with the RoT, for instance, reside on the same tamper-resistant chip. TRs outside the TCB may be realised as logical units in software. Note that the trust boundaries, especially involving TRs that are outside of the TCB, may be ephemeral. They may exist for some time for certain purposes, and then may cease to exist afterwards.

A general model process to extend the trust boundary beyond the TCB is verification. This is itself a TR implementing the verification process. We call this process and corresponding TR a verification entity, or verifier, to distinguish it from the process of validation of a TS by an external entity, the validator. Verification as a process to include a new component in the trust
boundary can come in essentially two flavours. First, and as a simplest option, the verifier measures a new component at the time of its initialisation. That is, the component, its status and configuration is uniquely identified. The result of this measurement is then stored. As an extension of this, the verifier can compare the measurements with reference values and decide whether or not to extend the trust boundary. That is, the verifier makes and enforces a policy decision. From the operational viewpoint, verification corresponds to predictability of the TS, as it can be assumed to be in a certain, pre-defined state after the verification process is completed. Validation, on the other hand, makes this property observable and therefore trustworthy. It means that a reporting entity transfers the results of verification to another party. The third, intermediate step performed by the reporting entity is that of attestation. Attestation is a logical consequence of verification and a logical precondition for validation. It is the process of vouching for the accuracy of measurement information, such that a relying party – the validator – can use it to decide whether it trusts the remote TS. For this, the measurement information must be bound to the specific TS and then be transmitted in a way that protects its authenticity. Verification, attestation, and validation are core concepts for operational trust which are tied to the lifecycle of a TS. This is detailed below.

A TS is owned by an entity (a person or other technical system) who or which is authorised to access certain TRs within the trust boundary, for instance the RoT. Ownership may implicitly be realised by physical possession of the TS, respectively, the platform containing it, or explicitly, for instance, by authentication of the owner through certain credentials. In the context of the TCG TPM specifications, the provisioning of such authentication data is called take ownership (TCG 2007a). An owner interacting directly with a TS is called local owner, whereas an owner whose communication with the TS is mediated in any way, e.g., through a communication network, is called a remote owner. When more than one TSS is contained in a TS, each may or may not have a different owner.

Figure 2 shows the separation of computing domains of TSS according to the TCG MPWG architecture (TCG 2008b). A TSS there consists of a dedicated Mobile Trusted Module (MTM), the hardware security anchor of MPWG specifications containing the mentioned RoTs, TRs (trusted resources and services in MPWG parlance), and normal software services and
components outside the trust boundary. The so called trusted engine in which all these reside is a secure computing environment, based on the RoTs providing, in particular, separation and controlled communication between different TSS. TSS can share TRs and even functions of MTMs with other TSS, conditioned by inter-domain validation and authorisation. Trusted engines, but also some of the MTMs, can be realised in software as long as at least one hardware MTM is present from which the RoTs of the soft ones are derived. Each TSS can be under the control of a local or remote stakeholder viz. owner. In the lifecycle of a mobile device, not all stakeholder TSS might be present. It is, therefore, necessary to define a process by which a (remote) stakeholder initialise the creation of a new TSS and take ownership of it. A variant of such a remote take ownership procedure is described below.

**Trusted Functional Building Blocks**

Special TRs of a TS are cryptographic capabilities which are usually within the TCB and may include one or more of the following:

- Symmetric and asymmetric encryption
- Hash value generation and verification
- Random number generation, e.g., with physical entropy sources
- Digital signature creation and verification

A TS may provide secure storage, i.e., places where, and methods by which data are protected from unauthorized access. For instance, cryptographic key material may only be used by a TS’ owner. A TS may have secure storage as a TR and for use by other TR within the TCB. As storage space there is commonly limited, general methods to extend secure storage have been envisaged, e.g., within the TCG standards (TCG 2007a). The secure storage within the TCB contains a RoT for Storage (RTS) for that, e.g., a cryptographic key. The RTS is then used to protect data outside the TCB, e.g., by encrypting them. A TS often has authorisation functionality incorporated to protect access to TRs. TPM authorisation is operated by storing 160 Bit secrets, e.g., password digests, within the hardware protected storage inside the TCB, namely the TPM chip.

**CONCEPTS BETWEEN TRUST AND ENFORCEMENT**

With the novel technologies of Trusted Computing, architectures of networked systems can be envisaged in which trust in nodes and terminals is scalable to a large extent. In this section, we develop a general model for interconnected TS, highlighting the main entities for trust establishment and the interaction processes between them.

**Establishment of trust**

Between trust and enforcement, the main bridging concept is separation of duties (Botha, R. A. and Eloff, J. H. P., 2001). It is genuinely thought of as a method to support certain protection goals in distributed systems, for instance privacy in client-server relations. A server may issue policies to a client who in turn enforces them using private local information, as described above. That is, separation of duties is normally understood as referring to duties on enforcement. But there is a natural relationship to trust. The relying party can delegate the enforcement to the other system only if it is operationally trustworthy. The establishment of operational trust between TS rests on the controlled exchange of information to enable observability and the pre-establishment of predictability. The latter can only be done outside of the TS.
Figure 3: Trust between platforms is mediated by organisational and technical methods.

Figure 3 shows a generic model exhibiting the role of external entities providing organisational assurance to TS. The security properties of a TS are rooted in the hardware trust anchor and the RoTs. These technical components cannot be examined while the system is deployed and operational. Therefore, they undergo a security evaluation during design and development. This is performed by an independent authority which, upon successful evaluation, issues certificates of security to the manufacturer of the security critical components. Apart from RoTs and trust anchor, this process may also comprise other TRs in the TCB, and involve different certification authorities. To ensure the homogeneous quality of evaluation processes and the different certification authorities, they are in turn assessed and certified by accreditation authorities, which may, for instance, be para-statal or private entities with state permits. The accreditation authorities can also serve to provide bridging information between certification authorities.

Certification authorities or technical entities informed by them, issue credentials to TS which are used by the TRs. These credentials, are certificates in the sense that they are verifiable in their integrity and provenance. A prime example is the Endorsement Key (EK) certificate issued to the TPM’s main RoT (the EK) by its manufacturer, as well as the Platform Certificate and other components’ certificates. These credentials and secrets derived from them by cryptographic means, are then also used in the interaction with external entities, in particular another TS. First, validation needs authentication and in many cases also confidentiality. Furthermore, secrets and credentials with trust inherited from the TS credentials are essential for operating system and trusted applications to build security associations, that is, channels which provide authentication, confidentiality, and integrity of communication. On top of security associations, applications within the extended trust boundary can build secure communication channels with well defined operational trust properties.

Trust establishment can hardly be effective without a mediator facilitating the various interactions just sketched. One key task of a mediation entity is to issue fundamental statements about the trustworthiness of a TS to another TS or relying party. Most importantly, the mediator
identifies the TCB (or selected elements, e.g., the trust anchor) as such, trusted and certified, component. To this end, the mediation entity needs to know the certificates issued by the certification entities, verify them when it receives it from a TS, and issue an according assurance statement to a relying party. It should be noted that validation is impossible without a mediator, if not all TS know the credentials of all other TS. Thus, mediation is, in fact, fundamental for validation. The best-known example for a mediator is the Privacy Certification Authority (PCA) defined in TCG standards, and described below in more detail. As we have seen, the role of a mediator between TS can extend further than protecting the TS privacy in validation processes. For instance, a mediator can also facilitate subsequent security association and secure communication, similarly to a CA in Public Key Infrastructures (PKI).

Though the model described above is generic and complete, this kind of trust provisioning infrastructure is not yet established in practice. Next, we exhibit main building blocks for trust establishment.

**Verification**

Verification is, in essence, a recording and controlling of state changes of a TS to the desired granularity. As such, it must be tightly bound to the operational cycle of the platform on which a TS resides, from initialisation to shutdown. Therefore, practical verification methods are mostly integrated with the boot process and operational cycle of platforms.

One general method for the internal verification of a TS is **authenticated boot**, and uses capabilities of the TCB to assess the trustworthiness of loaded or started software or hardware components at the time the TS is initialised, e.g., on power on. Authenticated boot is realised by starting certain functions of the RoT and the TCB before other parts of the TS. These parts operate as a **RoT for Measurement (RTM)**. This means that components that are started or loaded later on, are **measured**, i.e., they, and their status and configuration after start are uniquely identified, e.g., by forming cryptographic digest values over a (binary) representation of hardware component’s embedded code and loaded programs. According to the specific requirements, the measurement values may be stored in secure storage. Together with data necessary to retrace the system state from them, e.g., software names and versions, they form the **Stored Measurement Log (SML)** of the TS. On PC platforms, authenticated boot may include all components from the BIOS to the Operating System (OS) loader and the OS itself. One of the first proposals for authenticated boot procedures was the AEGIS system of Arbaugh, Farber & Smith (1997).

The most important existing realisation of authenticated boot is the one specified by the TCG. The system state is measured by a reporting process, with the TPM as central authority, receiving measurement values and calculating a unique representation of the state using hash values. For this, the TPM has several protected **Platform Configuration Registers (PCRs)**. Beginning with the system initialisation at power-up, for each loaded or started component a measurement value, e.g., a hash value over the BIOS, is reported to the TPM and stored securely in the SML, using the RTM. Concurrently, the active PCR is updated by an extend procedure, which means that the measurement value is appended to the current PCR value, a digest value is built over this data, and stored in the PCR. In this way, it is said that a transitive chain of trust is built containing all started and loaded components. As a single PCR stores only one value, it can only provide “footprint-like” integrity validation data. This value allows a validator to verify this chain of trust by recalculating this footprint, only in conjunction with the SML.
Secure boot is an extension of authenticated boot. It is of particular importance for devices like set-top boxes or mobile handsets that necessarily have some stand-alone and offline functional requirements. The common characteristic of devices equipped with secure boot is that they are required to operate in a trustworthy set of states when they are not able to communicate assertions on their trustworthiness to the exterior, e.g., before network access. In secure boot, the TS is equipped with a local verifier (a verification entity) and local enforcer supervising the boot process, which establishes the combination of a Policy Enforcement Point (PEP) and Policy Decision Point (PDP) to control the secure boot process. The local verifier compares measurement values of newly loaded or started components with Reference Integrity Measurement (RIM) values which reside in the TCB, or are protected within the TS by a TR, e.g., they are located in protected storage space, and decide whether they are loaded, resp. started, or not. Thus, the system is ensured to boot into a defined, trustworthy state.

Figure 4: Secure boot by local verification.

An embodiment of secure boot is described by the TCG MPWG (TCG 2008b). Initially, the RTM measures a software component (1) and creates a so-called Event Structure (2). An Event Structure contains an extend value, i.e., the actual result of a digest operation on the component’s code and extend data. As indicated in Figure 4, the RTM assigns the verification task to the RoT for Verification (RTV). Then the RTV uses the Event Structure (3) with the taken measurements (4) and verifies it against a set of available RIM (5). If the verification is accepted, the RTV extends the data to a dedicated PCR (6) and stores the Event Structure in the Stored Measurement Log (SML) (7). The SML contains the Event Structures for all measurements in the TPM and can be stored in any non-volatile storage, e.g., hard disk. Finally, the RTV executes the software component (8). The RoT for Reporting (RTR) is a dedicated secure element for a later validation. In the framework of policy enforcement, the RTM corresponds to a PIP which gathers information about the new component which lies yet outside the trust boundary. The RTV is a PDE/PEP combination. It is, therefore, responsible for evaluation of policies by matching measurements to RIMs, to allow or disallow component execution, and ultimately to make the new system state attestable by the RTR by storing it protected inside the TCB.
It is important to note a dual aspect of RIMs. On the one hand, they serve the local verification in a secure boot process. For that, they are complemented by a RIM provisioning infrastructure (TCG 2008b) that allows, for instance, updates of measured components, by provisioning of new RIMs to the TS. For an external entity to validate a TS after secure boot, it needs to compare the received event structure with stored RIMs and to verify the associated RIM certificates. Thus, RIMs and RIM certificates play an important role not only in verification, but also in validation.

Freshness of the attestation information is a key issue for validation (Guttman, et al. 2008). This necessitates extending the verification process from boot to operation time of the TS, which is a technically hard task in complex open systems. Such run-time attestation was first incorporated in IBM’s Integrity measurement Architecture (IMA, see Sailer, Zhang, Jaeger & van Doorn, L 2004). BIND (Shi, Perrig, van Doorn, 2005) is a method for fine-grained attestation of system memory that addresses only specific, definable pieces of code.

Validation

The mentioned separation of duties is also present on the process of validating a TS. Namely, based on the result of verification, the trustworthiness of the system may be assessed and, accordingly, policy decisions can be made in the validation. The separation of tasks in this process between TS and validator leads to three variant categories of validation. Before we introduce and compare them, we discuss one common base concept needed for any kind of validation.

Validation identities

A validation process of a TS must be supported by a validation identity which is exhibited to the validator. The validation identity must come directly or indirectly from a RoT, namely the RTR. As was noted before, validation is not possible without a mediator. This validation identity provider has the task to assert that the holder of the validation identity is a TS. Provisioning of a validation identity is an extension of identity provisioning in identity management (IdM) systems. The provider has to perform checks on credentials of the TS, including some or all TRs in the TCB, to assess if the TS is in a trustworthy state for validation. Furthermore, the provisioning of validation identities must be performed in a secure process, e.g., a security protocol on a dedicated secure channel. In case of remote validation, the validation identity may coincide with a global identity of the TS.

Validation using unique persistent validation identities is critical with regard to security. Validation may occur frequently and indiscriminately to many validators for varied purposes. Though the validation identities used may each not be easily associated to a user identity, they generally allow a tracing of the TS’ behaviour. Using the same validation identity for a group or all TS is not an option to resolve this for security reasons. Such a group identity would be a single point of attack/failure, that is, if one TS of the group is compromised, then all others cannot perform validation any more as well. The other option is to use ephemeral validation identities generated, for instance, once in each boot cycle, with determined frequency, or generated by the RTR for each validation.

Autonomous validation

Autonomous validation is a procedure whereby the validation of the TS does not depend upon external entities, and verification is assumed to have occurred before the TS will allow further communication attempts with the exterior or other operation. Thus, the verification process is
assumed to be absolutely secure in this case, as no direct evidence of the verification is provided to the outside world. The outside world makes the assumption that, due to the way in which TS are specified and implemented, a TS which fails verification will be prevented by its TCB from, e.g., attaching itself to a network or obtaining an authenticated connection to a remote entity. Autonomous validation lays all enforcement duties on the TS.

Autonomous validation is applying a closed, immutable system model to the TS, which is essentially the trust model used in smart cards. The TS verifies itself using the TCB, and the result is a binary value “success” or “failure”. Validation is then an implicit process by which the TS allows certain interaction with the exterior, such as network attachment. A typical example is the release of an authentication secret, e.g., a cryptographic key, by a smart card.

Security resting only on devices has been broken in the past and is more likely to be broken as, for instance, mobile devices become open computing platforms. Autonomous validation delivers little information for advanced security requirements: in particular if the TS is partially compromised, the exterior cannot gain any knowledge about its state. Labelling of rogue devices is therefore impossible, meaning that an exploit might proliferate without being noticed and cause significant damage to other stakeholders, such as network operators, before it can be contained.

Autonomous validation may be realized in such a way that verification is reactive to certain conditions, e.g., by not allowing certain functions, or by closing the device down and going to re-boot, depending on failure policy. This avoids network connection and seems advantageous. But this is also a vector for denial-of-service (DoS) attacks. The device must not attach to the network in a compromised state and, thus, has little chance to revert to a secure state. Remote management is also difficult, specifically there may be a loss of security in software download and installation since it potentially delivers values (software, secrets) to rogue devices. Thus, autonomous validation is prone to entailing out-of-band maintenance.

But also the validators bear additional burden since they have to keep track of the state of an autonomously validating TS. That is, if its state changes for instance by an externally forced update, this is only signalled through the next re-validation, which has no further informational content. It is the validators duty to update his database with the new TS state. If multiple validators can force updates on a TS, this may become complicated.

Finally, with autonomous validation, the freshness of the attestation data is not by itself guaranteed. For this security property to be fulfilled, autonomous validation would have to take place automatically on every system state change, strictly speaking. As autonomous validation happens infrequently in practice, e.g., during network attachment, the TS’ state may change significantly during operation of the TS, in a manner unobservable by the validator. Thus, an attacker may use this gap, for instance, to introduce malicious software. Autonomous validation is extremely prone to this kind of timing attack.

Remote validation

In remote validation, the validator directly assesses the validity of the TS based on the evidence for the verification he receives. The verification is only passive in this case, and the full SML must be conveyed to the validator. The model case for this is verification by authenticated boot and following validation. All policy decisions rest with the validator.

The main existing realisation of validation is one of remote validation. In a remote attestation, a TCG trusted platform exhibits SML and PCR, signed by an Attestation Identity
Key (AIK) to the validator. The AIKs are ephemeral asymmetric key pairs, certified by a Privacy Certification Authority (PCA) which acts as validation identity provider. More details on this process are found in (Leicher, Kuntze & Schmidt, 2009). The pseudonymity provided in remote attestation may not be sufficient in all cases. The TCG has additionally defined Direct Anonymous Attestation (DAA) (Brickell, Camenisch, Chen, 2004; Camenisch, 2004), which is based on zero-knowledge proofs (Chaum, 1985).

As both remote and autonomous validation are extremes of a spectrum of options which is subsumed in semi-autonomous validation, also remote validation has disadvantages. Remote validation, as represented by remote attestation, poses practical problems with respect to scalability and complexity, as it lays the full computational load for validation on (central) access points to networks or services. In particular, the validation of an SML can be very costly for platforms like personal computers with a large number of soft- and hardware components in numerous versions and configurations. This also requires an enormous database of RIMs, together with an infrastructure, to let stakeholders define the desired target configurations of TS'.

The same arguments make remote management of a TS, i.e., the controlled and validated change of configuration, impractical with remote validation. Furthermore, run-time verifications are desirable with remote validation, as otherwise only the state after boot is exhibited to the validator. The SML can be “withered” at time of validation. Thus, run-time verification becomes meaningless if it is not directly followed by validation, which would necessitate very frequent remote validations. Finally, remote validation of complex open TS’ compromises privacy, in spite of usage of a PCA, since the revealed SML might be almost unique to a TS. A similar, economic argument is the possibility of discrimination by remote attestation, that is, the threat that only recent versions of software of major vendors enters into RIM databases, forcing users of other programs to switch to these or loose service access. Some of the disadvantages might be alleviated by refined forms of remote attestation, such as semantic (Haldar, Chandra & Franz, 2004) or property-based attestation (Sadeghi & Stüble, 2004; Chen, et al., 2006), aiming at exhibiting the characteristics of components rather than a concrete implementation. These options, however, need more research before they may become practicable.

Semi-autonomous validation
Semi-autonomous validation is another procedure whereby the TS’ validity is assessed during verification within itself without depending on external entities, and policy decisions are made during verification. But in this case, the result of the verification and required evidence are signalled to the validator, who can make decisions based on the content of the validation messages from the TS. The signalling from TS to validator must be protected to provide authentication, integrity, and confidentiality if desired. A model case for semi-autonomous validation is secure boot, followed by a signalling of the event structure and indication of RIMs to the validator. Semi-autonomous validation symmetrically distributes verification and enforcement tasks between TS and validator. Specifically, in secure boot, the former makes decisions at load time of components, while the latter can enforce decisions on the interactions permitted to the TS upon validation, based on the state evidence provided.

Semi-autonomous validation may be a promising avenue to a remedy for the disadvantages of the other two options. It can potentially transport the validation information more efficiently in the form of indicators of the RIMs used in verification. This can also be used to protect privacy, for instance, when such an indication designates a group of components with the same functionality and trustworthiness (such as versions). This would be similar to semantic and
property-based attestation, and it is conceivable that semi-autonomous validation may be combined with the mentioned advanced forms of remote validation. The interplay of enforcement in verification during validation on the part of the validator, also opens options for remote management of a TS.

On the path to technical realisation of such opportunities, the Trusted Network Connect (TNC) working group of the TCG has introduced the concept of remediation (TCG 2008c), to obtain “support for the isolation and remediation of ARs [Access Requestors] which do not succeed in obtaining network access permission due to failures in integrity verification.” (p. 24). This allows, in principle, “to bring the AR up to date in all integrity-related information, as defined by the current policy for authorization. Examples include OS patches, AV [Antivirus] updates, firmware upgrades, etc.” (p. 25). Concrete concepts for realization of remote management will have to rely on an infrastructure for the efficient representation and communication of RIM information. TCG MPWG has started to define such services for mobile TS (TCG 2008b), in particular to ingest RIMs for verification. The Infrastructure Working Group of the TCG is establishing a generic architecture and data structures verification and validation (TCG 2006). More research and development is needed to devise efficient and effective semi-autonomous validation on this path.

It is important to emphasise the role played by RIM certificates in semi-autonomous validation. RIM certificates are provided by a certification authority which has assessed, directly or by delegation, the corresponding TR. Certification methods and bodies can be diverse and lead to different levels of operational trustworthiness. This leads to further flexibility for a semi-autonomous validator who gets more fine-grained information on the TS.

Semi-autonomous validation is also the only practical validation option for systems which are resource limited so that a) they lack the processing capabilities to do autonomous validation, and b) lack the memory and/or communication capabilities to perform the extensive reporting needed for remote validation. Shaneck, Mahadevan, Kher & Kim, 2005 give an example in the context of Wireless Sensor Networks, in which both limitations hold for the sensor nodes. Their proposal is to send memory probing code to the sensors that calculate a digest value of the static memory content (code and parameters) which should lead to a predictable result which is returned to the base station for validation. An attacker could obviously try to circumvent this “attestation” by using saved, original memory contents to produce the correct outcome. As long as this attack is performed on the sensor itself it will, however, inevitably lead to delays which can be enhanced by randomisation, self-modifying probing routines, and obfuscation methods. Thus, if a significant delay in the sensor’s answer occurs above a certain threshold, the sensor is invalidated.

Validation and enforcement
Validation and verification are the central conceptual link between trust and enforcement, because based on the results of validation, various policy decisions can be made, and verification in turn can incorporate enforcement during secure boot. It is instructive, though it does not add technical content, to map the concepts of validation in the three variants described above, to the basic architecture of enforcement systems.
Figure 5: Mapping Validation variant to policy enforcement

Figure 5 shows a simplified picture. A common trait of all variants is that the TS needs a PIP as minimal resource to support the validity decision by the validator. The PIP has to perform the measurement of the TS’ state for this and to securely record the results using the RTM. Since validation is always performed for a purpose, there is, in all cases of practical relevance, a PEP present at the part of the validator. Based on the attested information, it can enforce decisions such as granting network access. The richness of the latter information varies significantly between the validation variants.

In remote validation, the TS has no other means to build trust with the validator than to transmit the full SML to the validator, plus information binding it to the TS state and protecting its authenticity (e.g. PCR values signed by the RTR). The validator’s PIP has to contain a database of possible allowed TS states including reference measurement values. Based on the attestation and the state reference information, the PDP at the validator re-traces the SML (e.g. re-calculates the digest values). The PEP obtains a graded result from this process stating up to which position in the SML the TS was in a good state. On this information, the PEP acts for instance by (dis-)allowing network access.

Autonomous validation is the other extreme. All functionality for measurement, verification, and enforcement during secure boot and runtime are localised in the TS’ PIP, PDP, and PEP, respectively. No explicit attestation statement is made to the validator who has to rely on the implicit signal that can be inferred, e.g., from an authentication attempt. The validator’s PEP can enforce only policies based on the static information contained in this signal, e.g., system type or identity. Since validation information is not present, no validation-specific PIP and PDP are used at the validator (however, a non-validation PIP and PDP can be constructed in this case based on TS identities, and connection history – in effect a traditional authentication, authorization and accounting [AAA] system, see de Laat et al., 2000).

Semi-autonomous validation allows for equally capable policy systems on both sides. The key to this is a codification of attestation data. It need not be transferred as a complex SML including measurement values of all components. It is replaced by a concise event log containing essentially references to RIMs, respectively, associated certificates (the precise content may depend on implementation requirements). This abstraction is made possible by the PDP in the TS which, at the time of verification, e.g., during secure boot, makes the association of component to target RIM. For that, it relies on an internal, protected, RIM database, whose management adds to
the functional role of the PIP (beyond measurement). Attestation to RIMs allows interaction with the validator in validation. The PDP of the validator can use its own RIM database (provisioned by its PDP) to compare the attested TS state with fine granularity to a desired state. The PEP communicates the outcome to the TS and can thus initiate i) provisioning of new RIMs to the TS, ii) unload of undesired components, iii) load of new, desired components, and by that finally iv) updates of components. These processes are captured by the term remediation. To show the success of the remediation, the TS needs to revalidate only using the newly adjoined part of the event log. From the viewpoint of policy systems, RIMs add an essential piece to enable general policies for validation: A codified ontology on which conditions can be evaluated and decisions be taken.

**Remote take ownership**

The lifecycle of a trusted system begins with a conformance and compliance testing of its trust anchor and TCB. In deployment, there must always be at least one TSS present for the device to be usable. As was noted before, in multi-stakeholder scenarios, there should be a way to obtain new TSS and assign them to an owner by a well-defined, trustworthy process. We give a simple example for a possible remote take ownership (RTO) procedure adapted from (Schmidt, Kuntze & Kasper, 2008) using the facilities of the TCG MPWG specifications. This process is rather generic, does not depend on the physical presence of the remote owner and can, for instance, be executed over-the-air. The involved TSS are those of User (U), Device Manufacturer (DM), Remote Owner (RO), the latter of which is initially not present. Figure 5 shows the process.

![Figure 5: Installation and initialisation of a TSS on behalf of a remote owner.](image)

U requests the RTO for an RO TSS from the DM TSS, for instance, by selecting a service provider. The DM TSS is equipped with a TR which allows the pristine boot of another TSS, i.e., the installation and initialisation, including secure boot into a pristine, unowned, but otherwise functional state. The pristine RO TSS can thus perform local verification and report corresponding validation data to RO DM. RO DM then issues an RTO request to RO and after RO acknowledges it, a validation of DM TSS and the pristine RO TSS is performed by the RO. If the RO has convinced himself of the trustworthiness of both TSS', he generates unique
credentials to individualise the RO TSS. Along with other data, such as security policies and other secrets, the credentials are signed by the RO and sent to the pristine RO TSS via the DM TSS. RO TSS updates its RoTs with the new credentials received and installs the other data. It should be noted that the indirect communication between RO and RO TSS is carried out over a secure channel, which was established previously either with the RTO request to the RO, or in validation. This means that even the DM TSS has no access to the secrets transferred to the RO TSS. The secure channel is based on the credentials, e.g., asymmetric key pairs, of the RoTs of the pristine RO TSS which are generated in pristine boot. In the last step, the status message toward the RO is also an indication that the RO TSS has passed local verification, i.e., this message constitutes semi-autonomous validation. In (Schmidt, Kuntze & Kasper, 2008) we also showed how a TSS can be migrated between devices while maintaining ownership.

APPLICATIONS

One of the first applications of trust technologies was a mapping to policy enforcing systems, in particular in IRM and DRM systems, to endow the consuming client with technical security measures (Sailer, Jaeger, Zhang & van Doorn, 2004; Sandhu, Zhang, Ranganathan & Covington, 2006). TCG technology also applies to the communication with a client consuming data as we have demonstrated in (Kuntze & Schmidt 2007a) and (Schmidt & Kuntze 2009). Trust technology also applies to contextual information used by PIPs in enforcement. A good example is the location of a mobile device for which we have described a system architecture and protocols in (Schmidt, Kuntze & Abendroth 2008). A key idea bridging enforcement and trust is, in all these cases, to endow the PIP with verification information and validation tasks.

Here we outline three different applications demonstrating the synergies between operational trust and security. They demonstrate the use of trust technology in the relationships of a) users and systems, b) information systems and data, and c) autonomous systems and communication networks.

Trust in identity management systems

As a platform-neutral security infrastructure, TC offers ways to establish trust between entities that are otherwise separated by technical boundaries, e.g., different access technologies and access control structures. Not surprisingly, some concepts of TC are rather similar to Identity management (IdM) and federation. We have explored this relationship in various directions in (Kuntze, Mähler, Schmidt, 2006; Kuntze & Schmidt 2007b; Fichtinger, Herrmann, Kuntze & Schmidt, 2008). As one important example, let us here consider Ticket Systems along the lines of (Leicher, Kuntze & Schmidt, 2009). In a ticket-based authentication and authorisation protocol like Kerberos (Steiner, Neuman & Schiller, 1988; Neumann, Yu, Hartman & Raeburn, 2005) software tokens are used to prove the identity of a single entity. Based on these tokens, access to certain systems is restricted to entities producing appropriate tokens. Additionally, data embodied in the token can also be used to implement an authorisation control, enabling a token based access control scheme beside the mere authentication. These tokens are an electronic analogue to physical tickets.

The Kerberos concept relies on two main servers, Authentication Server (AS) and Ticket Granting Server (TGS), which both issue the tickets. Each ticket contains two parts: one part is encrypted for the next target server and thus cannot be decrypted by the client. This part contains a session key that will be used by the next communication step. The session key is encrypted for
the client in the second part of the ticket. If the client can decrypt this part, he obtains the session key to request the next ticket. There are four main steps:

1. Request and receive the **Ticket Granting Ticket (TGT)** from the AS, store it for future use. Decrypt the session key to encrypt the authentication data for the **Service Ticket (ST)** request.
2. Send the TGT, together with the encrypted authenticator to the TGS.
3. Receive the ST from the TGS. Decrypt the new session key to encrypt the authentication data for the service request.
4. Send the ST, with the encrypted authenticator to the service provider.

The main goal of the **Trusted Kerberos protocol** is to show how TC can be used to enhance a Kerberos. The design has two targets: enhance the security by providing means to bind the tickets closely to the user’s TS via the TPM, and secondly, protect the privacy of the user accessing a service. A proof-of-concept of trusted Kerberos ticket system has been realised, based on the ethemb TC emulation framework (Brett & Leicher 2009).

With respect to security in particular, a duplicate of the ticket cannot be created and used on another system. In addition, there is no client secret shared between the Kerberos AS and the client. This is one weak spot in the standard protocol, as it allows eavesdroppers on the network to collect TGTs and try to decrypt them offline using brute force and dictionary attacks. Mostly, this password is chosen by the user, thus the chance of finding weak passwords if there are lots of clients in a Kerberos realm makes it easier for attackers. The impact of such an attack can become quite heavy. As the TGT represents the user’s identity in the Kerberos realm, the attacker will gain access to all services the legitimate user has. By using one-time passwords that are cryptographically strong and can only be decrypted using the target TPM, the offline attack on captured tickets becomes impracticable. The passwords can be generated on the server side, thus the system does not have to rely on the computational power of the TS. As a second point adding to the security on the server side, there is validation of the user’s system included. When the user wants to acquire an ST to access a service, he is challenged by the TGS to remotely attest system conformity. This process relies on the IMA concept (Sailer, Zhang, Jaeger & van Doorn, L 2004).

As a result, only clients in a certified system state will be able to access the service. Figure 6 shows the trusted Kerberos protocol. The TGT is augmented by a **TCTicket** containing the claimed identity of the TGT and service request, and which is signed by the TPM.

![Figure 6: Trusted extension of the Kerberos protocol.](image-url)
Privacy is protected by separation of duties between the TGS and the AS. In the presented concept, a user has to register with one AS, revealing his complete identity to the AS. Then, the user can register multiple partial identities, referred to as claimed identities. The user claims to be in possession of the identity and the AS can certify this by providing a certificate. The AS is the only instance being able to map the claimed identities to the real identity. To further enhance the privacy, no communication shall reveal information to parties other than the current communications partner. During TGT requests, communication between the client and the AS, the client’s real identity information is protected as it is encrypted using the AS public key. The response from the AS is encrypted using the public EK of the TPM. When the client uses a TGT to request a ST from the TGS, the communication is encrypted by the session key from the TGT. The TGT can only be decrypted by the TGS, revealing the session key and allowing the TGS to decrypt the data provided by the client. The response is cryptographically bound to the TPM by encrypting the session key with a one-time key bound to the Certified Signing Key (CSK) (Kuntze & Schmidt, 2007b) from the TCTicket and secured by the TPM. Only the user in possession of the TPM and the credentials for the key usage in the TPM can decrypt the ST and, thus, use it to access a service. In the concrete embodiment, AIKs are used as one-time credentials in the TGTs, and the AS plays the role of a privacy CA. Thus the TGT request/receive process is realised as a variant of AIK certification.

Even the request to the service provider reveals no information concerning the user’s identity to someone else. Only the targeted service provider will be able to decrypt the given partial identity. Due to the separation of duties, the service provider will not gain information about the real identity of the user. As he is associated to a TGS that issued the ST, a provider can contact the TGS operator in the case of misbehaving users. The TGS can then forward the message to the corresponding AS for this identity. As the AS has a mapping of the claimed, partial identities and the real identities, the user can be made liable. In addition, the service provider is not required to keep a database of existing users. So, he can register and associate with a TGS (via legal contracts), enabling multi-service single-sign-on experience for users wanting to access multiple services.

**Trust for transactions in workflows**

There are various instances in intra- and inter-organisational workflows in which digital documents have to be changed in significant ways. At the ingestion, for instance in a (semi-) electronic postmaster system, incoming documents are scanned or transformed into a uniform data format for internal processing. In the administrative process, documents are edited, extended, and other documents are spawned from them, before they are stored in an internal data warehouse for short-term use and/or an (external or internal) archiving serviced for long-term preservation. Before conveying documents to an external party, they are still, in many cases, printed for final authorisation. All these processes are increasingly supported by workflow systems. The components of such systems are organised according to the enforcement paradigm of IRM. Trust in the components of a workflow system which perform the mentioned transactions on digital documents is generally lacking. This has a significant negative impact on the auditing of such systems and, ultimately, on the probative force of the documents produced.

Schmidt & Loebl (2005) propose the concept of a transformation seal to incorporate validation data of a workflow system. This was later realised for systems performing transformations on documents (Schmidt, Kreutzer & Accorsi, 2007). The seal yields to forensic inspection many details of the process. This information has probative value due to its binding to
the new document via the seal’s signature. This is sufficient in many cases, e.g., when the workflow system is secured by organisational measures or physically. However, in general distributed processing, this may not be the case and the question of the trustworthiness of the workflow system becomes urgent. This problem emerges in three main application fields: First, if the workflow system is open and weakly secured; second, if the transactions are performed in a distributed fashion; and third, if the party performing the transaction has its own, specific interest in the result of it and has the system under their control. There are three requirements on attested transactions, i.e., seals obtaining secure, auditable information on the system on which the process is performed:

A. **Binding to the technical system and validation.** The system and its current state during the transaction must be uniquely identified.

B. **Binding to the transformation process.** The mentioned information must be bound to the particular transaction.

C. **Binding to the target.** The mentioned information must be bound to the new document after the transaction.

A TS able to perform verification and validation is a conceptual prerequisite for attested transactions. It is straightforward to devise a structure which extends transformation seals to yield the bindings A-C. This is shown in Figure 7. During processing in a single step of the transaction, the system collects state information, for instance the SML, and adds them to the transaction report. The TS also adds a security digest, e.g., PCR values to this data for later validation. The single steps, as well as the whole transaction report are then secured by machine signatures, e.g., based on AIKs as described in (Kuntze and Schmidt 2007a). The outermost of these signatures establishes the binding to the converted contents. The seal is completed by a signature of the responsible party.

![Figure 7: Information structure of a seal for attested transactions on digital documents.](image)

The method presented above is certainly more generally applicable. It has been extended to BEPL-controlled workflows to produce what we call trusted process slips by Kuntze, Schmidt, Rudolph & Velikova (2008).
Home NodeBs and machine-to-machine communication

The major industrial standardisation group for mobile communications, 3GPP, currently studies two advanced applications for their specification Releases 8 and 9, which pose specific security requirements. The latter are considered in the security working group (3GPP SA3 2009a). Both applications have in common that devices are

i) no longer considered as closed, immutable environments for the storage and handling of sensitive data, as mobile handsets have been traditionally viewed, and

ii) these special devices are under the control of a stakeholder different from the mobile network operator (MNO), and are connected to the core network only intermittently and over an insecure link, in general.

The first application regards so-called Home (enhanced) nodeBs, short H(e)NBs, better known as femtocells. These are small, portable access points to 3G networks which are generally placed on the premises or in the homes of stakeholders called Hosting Party (HP). The Hosting Party becomes a mediator for mobile communication and services in a small, designated geographic area. This can be used to provide mobile services in hitherto inaccessible areas (due to bad radio conditions) such as in-house or factory environments. It is also an option for private households and the SOHO sector, as a H(e)NB can be a unified access point to broadband Internet and mobile networks. MNOs thus view H(e)NB as a very interesting new market they want to control.

In H(e)NB usage scenarios three stakeholders, Users – HP – MNO, are related by Service Level and usage agreements. The H(e)NB stores a lot of sensitive data in this context, such as the Hosting Party’s authentication data, embodied, e.g., as a mobile network subscription, the list of User Equipment (UE) which is allowed to connect to the H(e)NB, stored as a Closed Subscriber Group (CSG), and an Access Control List (ACL). Some of this data can be private to HP and/or Users. Also, the location of the H(e)NB needs to be controlled to protect the mobile network from interference and prevent illegitimate extension of services.

The general scenario for the communication between an H(e)NB, UE, and operator core network is shown in Figure 8. It introduces two new network entities, one tasked with security, the other with servicing of the H(e)NB. The Operation, Administration and Maintenance (OAM) is a function in the backhaul of the core network which provides remote management functionality to the H(e)NB, in particular, software downloads and updates, setting of radio and other parameters, etc. The Security Gateway (SeGW) is the main entry point for H(e)NBs into the operator’s core network, and its main purpose is to protect the network from illicit connection attempts and any kind of attacks that may emanate from rogue H(e)NBs or an attacker impersonating an H(e)NB.

Figure 8: Generic H(e)NB communication scenario.
The second intended application regards general machine-to-machine (M2M) communication. Typical examples for M2M Equipment (M2ME) are vending and ticketing machines. More advanced scenarios comprise, amongst others, remote metering of combined heat and power plants, machine maintenance, and facility management. Some of these are described in (Kuntze and Schmidt, 2006a & 2006b). If M2ME are connected to backed systems via a mobile network, MNOs will be enabled to offer value-added services to M2ME owners, beginning with over-the-air (OTA) management. Like H(e)NBs, M2ME are under the control of a stakeholder different from the MNO, and who has certain security requirements which may be different from the MNO’s. Security of H(e)NB and M2ME are studied in (3GPP SA3 2008a & 2008b), respectively. The respective threats, risks, and ensuing security requirements are similar in both cases, We consider H(e)NB as those are formulated more concretely. Threats can be grouped into six top-level groups.

1. **Compromise of Credentials**, comprising brute force attacks on tokens and (weak) authentication algorithms, physical intrusion, side-channel attacks. And a malicious hosting party cloning an authentication token.
2. **Physical attacks**, e.g., inserting valid authentication token into a manipulated device, booting with fraudulent software (“re-flashing”), physical tampering, and environmental/side-channel attacks.
3. **Configuration attacks**, are fraudulent software update / configuration changes, mis-configuration by the HP or user, mis-configuration or compromise of the ACL.
4. **Protocol attacks on the device.** These attacks threaten the functionality and are directed against the hosting party and the users. Major examples are man-in-the-middle attacks upon first network access, denial-of-service (DoS) attacks, compromise of a device by exploiting weaknesses of active network services, and attacks on OAM and its traffic.
5. **Attacks on the core network.** These are the main threats to the MNO; Impersonation of devices, traffic tunnelling between them, mis-configuration of the firewall in the modem/router, DoS attacks against core network. In the case of the H(e)NB, it also regards changing of its location in not allowed ways. Finally, this includes attacks on the radio access network using a rogue device.
6. **User Data and identity privacy attacks** include: eavesdropping of the other user’s UTRAN or E-UTRAN access data; masquerading as other users; user’s network ID revealed to H(e)NB owner; masquerade as a valid H(e)NB; provide radio access service over a CSG.

The core functional requirements which are new for both H(e)NB and M2ME, regard mainly the authentication of the different stakeholders, and the separation of functions and data between them, i.e., domain separation. In particular, the authenticity of the HP or M2ME proprietor shall be made independent of device authentication to the network. Furthermore, secret data of the HP must be protected from access by another party, even the MNO. The device has to perform security-sensitive tasks and enforce security policies towards both the access network, and the connected UE. This must be possible in at least a semi-autonomous manner, to provide service continuity and avoid unnecessary communication over the backhaul link. Another important security area is remote management by OAM or OTA, respectively. The device needs to securely download and install software updates, data, and applications.

The need is to separate the authentication roles while concurrently minimising changes to the core network and, thus, to re-use standard 3G authentication protocols such as EAP-AKA, described by Arkko, J. and Haverinen, H. (2006). The approaches discussed so far in the standardisation group envisage separate authentication bearers for the HP, respectively M2M
owner. They could be embodied in a so-called HP Module (HPM) in the former, or in managed identities (MIDs) in the latter case. Both might just be pseudonyms for UICCs, i.e., 3G SIM cards. Various security concerns have been raised against usage of removable smart cards in the M2M case. On the other hand, maintenance operations necessitating exchange of such smart cards, e.g., for updates or operator change, is to be avoided, as it would be very costly for a large fleet of geographically dispersed M2ME. Another option cautiously considered recently is the download of AKA credentials to a secure environment in the device. We have previously described a scheme using genuine TC technology allowing for this option (Schmidt, Kuntze & Kasper 2008), called a virtual SIM. As of yet, MNOs fear the potential ease of churn, and smart card vendors fear a loss of continuous revenues by such advanced technology.

In any case, the security requirements and also advanced OTA or remote management, require particular security features on M2ME and H(e)NBs. This has led to the inception of a concept called Trusted Environment (TRE), which is nothing but the TS of our parlance. Minimal requirements on such an environment already use a lot of concepts of TS, in particular verification, secure boot, validation, and remote ownership.

A TRE needs to securely interact with other parts of the system. It is interesting to look at these TRE interfaces, as they are a general model for how the TCB of a TS communicates with the rest of the platform. Basically, all TRE interfaces are initialised in the secure start-up process of the TRE, and are thus assumed to operate correctly. There are two broad security categories of TRE interfaces:

1. Unprotected interfaces. These interfaces let the TRE with general resources of the device which are not assumed to be secured against tampering and/or eavesdropping. Even unprotected interfaces may also benefit from other security measures such as data encryption, or making the interface available only after the TRE checks the code of its counter-part resource across the interface, for example, during a secure boot.

2. Protected interfaces: These interfaces provide either protection of the integrity and/or confidentiality of the data carried across them, using either security protocols or secure hardware. If security protocols are used, they may also provide authentication, and message authentication and/or confidentiality.
In the design, various aspects are relevant for the choice of a particular TRE interface configuration. Unprotected interfaces may be chosen when the communicating entity does not provide protection of the communicated data. Protected interfaces may be chosen when there is a need to provide protection of data integrity and/or confidentiality between the TRE and another resource that the TRE needs to communicate with. Accordingly, the capabilities of the TRE may vary. Figure 9 shows one proposal for a TRE within an H(e)NB and what other resources it might connect to. This is a minimal configuration, containing essentially the capability to compute and send to the SeGW the parameters needed for device authentication of the H(e)NB, functions for H(e)NB validation, including code-integrity check of the rest of the H(e)NB at boot time, and minimal crypto capabilities (a true random number generator). With regard to authentication, it is envisaged that TRE might logically contain the HPM.

**FUTURE RESEARCH DIRECTIONS**

The most important applications of trusted systems lie in loosely connected, heterogeneous nodes communicating via convergent communication networks. Within the European Union’s seventh research Framework Programme, major efforts are made to meet the challenges of such networks, which are known under the summary term “future internet” (FI), (European Commission 2008). FI is characterised by distributed storage and processing of data. Network nodes are not only used for data transmission and terminals not only for applications. Nodes become hosts for data and application services while terminals participate in an ad hoc, more or less controlled manner, in data transmission through the network. This is a kind of convergence that transcends every layer of the network and it goes hand-in-hand with the convergence between mobile communication networks and the core Internet.

FI security deals with a system where core capabilities, as well as application overlay networks, operate in a distributed way and – by themselves – with little functional guarantees, for
instance, with regard to Quality of Services. Thus, the main challenge to guarantee security and resilience of the network in this environment is to establish a uniform foundation for trust in nodes and terminals, as well as security of communication between them. A trust foundation must establish RoTs for connected devices relying on hardware security to reach the necessary assurance level and make security scalable. This calls for a broad-scale application of the methods described in this chapter. In a Future Trusted Internet, we envisage three main areas of protection:

1. **Protection of nodes and devices against misconfiguration and malware.** As most attacks to the Internet infrastructure and services are launched from within – by collections of rogue nodes such as bot-nets – a node or terminal in the Future Internet should be able to bring itself into a trustworthy state and perform validation.

2. **Protection of user credentials and Privacy-friendly ID-management.** As more and more user data is processed on distant nodes not under local user control, user data, as well as his access and even personal credentials, travel over the network and require appropriate protection. The network design needs to enable the control of personalised data in a way that implements a thorough ‘need to know’ principle.

3. **Trust in processing and probative value of processes and communication.** FI evolves from an information transmission to an information processing facility, and bears a great deal of critical business processes. Service oriented architectures’ users, for instance, heed little care about where and how their precious data are processed. They naturally assume that it is protected. New requirements stem from the necessity, nowadays implemented by global treaties, EU, and national jurisdiction almost globally, to make every operation in a business process auditable and to keep detailed, and true, records. Therefore, the FI must provide ubiquitous support for non-repudiation not only of communication, but also of distributed data processing.

To realise the vision of the protection areas, there are main research challenges to be tackled:

1. Find methods to leverage the hitherto scattered and unconnected, hardware-based security building blocks into an overarching trust architecture to protect connected TS. This comprises new, high-performance, methods for application separation, perhaps beyond virtualisation. Furthermore, security primitives need to be developed that enable fundamental operations on, and communication with TS.

2. A uniform credential management architecture leveraging hardware trust must be envisioned. Trusted means to control operations with user credentials, such as enrollment and migration between devices, shall ensure users’ privacy and seamless access to network services. Also needed are novel, secure, multi-modal, seamless user authentication methods, and methods to validate a TS to the user.

3. New and uniform methods for the provisioning of security to higher application levels must be deployed throughout the FI. This enables applications to validate the state of the host system, and in turn, the host systems to provide scaled security to each application. This security comprises, in particular, new means to bind trust-related information to data that is stored or communicated to other nodes. Those methods need to be independent of data formats and representations.

As we see, the main thrust of research, coming from the impact of scalable trust concepts, is an applied one. In the real-world scenarios which are the empirical basis of computer science, appropriate security concepts cannot be designed without taking the value of information into account. This is the most important ancillary condition determining scalable trust concepts for
various purposes. Such applied research must be accompanied by, and integrated with, a broad interdisciplinary approach that emphasises the user perspective. Thus, realisation of trust to make future information networks such as online communities (Boyd & Ellison, 2007) needs input from sociological, economic (Kollock, 1999), psychological, and legal disciplines.

ACKNOWLEDGEMENTS
The authors thank Shigou Lian, Ilaria Matteuci, Yogendra Shah, and Davide Vernizzi for valuable suggestions and comments.

REFERENCES


A word on nomenclature. In the present chapter, we introduce a unified wording centered on operational trust. According to this aim we deviate from the often used and often confounded notions of “trusted system” and “trustworthy system.” The National Security Agency (NSA, 1998) defines a trusted system or component as one "whose failure can break the security policy", and a trustworthy system or component as one "that will not fail." The former notion is most closely related to our definition of a TS in the operational sense, more precisely a TS endowed with a particular enforcement task in a certain operational context. The notion of trustworthy system makes little sense, operationally, and relates to the TCB, at best. For more discussion see Anderson (2008, Chapter 1).

This again deviates from the literature, where mostly verifier is a receiver of some information which can be computationally matched to yield a binary answer to a question related to the security of a system. In particular in semi-autonomous validation, which we argue is the practically most important case, this function is internalized in the TS. This justifies the introduction of the term validator to denote the external entity which ultimately assesses the operational trust in a TS based on the verifier’s information. “Attestation” in turn is, as we explain, the process of securely (protecting data authenticity) communicating with the validator. Remote Attestation is just one embodiment thereof.

This somewhat clumsy terminology of 3GPP refers to „nodeB“ which designates a mobile network’s base stations in 3G, e.g., UMTS networks, and to the Long Term Evolution Project (LTE) of 3GPP. According to LTE conventions, entities beyond Release 9 are to be differentiated from their earlier counterparts by an (e) for “enhanced”. In the case of nodeBs, the enhancements consist in additional functions taken over by the base station beyond providing radio access to terminals. For instance, Home (e)nodeBs may function as Internet modems and wireless access points.