

# Trusted Computing or *The Gate-keeper*

Klaus Kursawe and Christopher Wolf  
{Klaus.Kursawe, Christopher.Wolf}@esat.kuleuven.ac.be  
chris@Christopher-Wolf.de  
K.U.Leuven, ESAT-COSIC  
Kasteelpark Arenberg 10  
B-3001 Leuven-Heverlee, Belgium  
<http://www.esat.kuleuven.ac.be/cosic/>

## 1 Motivation and Introduction

Recent years have seen serious attacks on the backbone of our information society infrastructure. To mention a few, we want to draw the attention of the reader to the disastrous consequences of the Slammer worm which disconnected South Korea from the Internet, knocked out ATMs, and caused an estimated damage of 7.7 billion USD for removal and down-times. On the other hand, Internet commerce grows, and every year more and more people use the Internet for business to consumer transactions. Forrester's overall estimations for 2005 and 2006 in this segment are 8.8 billion USD and 12.3 billion USD. However, the consumers' confidence in the use of the Internet will drastically decrease if incidences like Slammer become more frequent. Another issue is the use of computers as a replacement of paper based voting machines, *e.g.*, in the USA or in India. Obviously, the overall outcome of these elections drastically effects at least the countries in question, and is hence a valuable asset which needs to be protected. Apart from the outcome itself, any serious doubt in the correctness of the result will shake the trust people have in their own political system. Again, the need for machines which do what they are supposed to do and nothing else is high. Finally, both the music and the film industry call for means to protect their property from being copied and distributed virtually for free, while users are getting increasingly worried about their private data being collected and electronically processed (or at least, they should).

All these items seem to be rather unconnected, but may be solved or at least eased with one development: trusted computing. The idea is rather simple: instead of trusting the correctness of software, *e.g.*, the operating system and the applications, we move the level of trust to a specific piece of hardware.

From a philosophical point of view, this "change" does actually not change anything: basically, we just move the reason of our trust to some specific token — being either software or hardware. At the moment, we have to trust in some specific software, which includes the operating system and all hardware drivers running on the machine in question. In the future, we need to establish trust in a hardware device. Before moving to a description of possible hardware devices in the next section, we start with a short introduction of the software core of trust.

### 1.1 The Caretakers

By now, all software based solutions need to trust in the core of each software, *i.e.*, the operating system. Using the picture of a large house, *e.g.*, in a university, we see that there are plenty of tasks to fulfil, and that many people work in many rooms. Once in a while, items are broken and need replacement, or just simple maintenance like, *e.g.*, cleaning. Hence, the care-takers have keys to all rooms and can go there at will. In the "real world" of a computer, the care-takers represent the different parts of the operating systems while the members of the university are the picture used for the different applications, and the rooms are the memory-protected areas of the computer. We see immediately that there are few things to do without these "care-takers" in a computer: to type a text, a word processor needs input from the keyboard and the mouse — both of them given to the word processor by two care-takers, namely the mouse and the keyboard driver. As soon as a key-stroke is received, the corresponding character needs to be displayed — which again involves the work of a care-taker, in this case the driver of the graphical card and the display. At this

point we did not talk about saving the text or printing it. Again, this would involve the work of even more care-takers.

As we saw, the care-takers and hence drivers are very important for the correct working of a nowadays computer. In particular, they are necessary to ensure that the same programmes run on totally different hardware platforms and that we can easily change the configuration of our computers, *i.e.*, by plugging in a faster Ethernet card or a printer with higher resolution.

In the real world, each care-taker would have very limited power and can only do what is necessary for the fulfilment of his tasks, *e.g.*, the gardener has access to the green house, but not to the exam locker. In the computer-world, things are unfortunately different: all care-takers have equal rights and hence, can perform each others' tasks. In the picture of the university this means that *any* care-taker has the keys to *all* rooms and may go there at will. While we would refuse such an arrangement in any real-world situation, this is the case in our current computer architecture — and holds for Windows, Linux, and MacOS, as all of them use a large, monolithic kernel. To come back to the chain of trust we introduced previously: we actually need to trust *every single care-taker*, *i.e.*, that this person cannot be bribed or does not lose its key or may be dishonest by nature. Translated to the computer world, this means that every single driver, in addition to the core of the operating system must be trustworthy: it may neither be a Trojan horse, *i.e.*, performing a task different from what it claims to do, nor may it have a security critical bug, *e.g.*, allow the execution of code (*e.g.*, due to a buffer overflow). Given that we expect one security critical bug in 1000 lines of code, and the size of modern kernels (the Linux kernel in version 2.4, for example, has about 1.5 million lines of code), those operating systems are not suited for any serious security applications. Hence, assuming the security of these kernels is a rather big assumption and given our expertise of the past years, a void one.

## 1.2 The Gate-house

Hence, to overcome the problems outlined in the previous section, we need two things: first more separation, and second a trusted core on which we can base our trust in a specific computer. As we saw, this is a very risky business by now, and the incidences of the previous years have shown that our doubts in the trustworthiness of nowadays operating systems are not only an academic exercise.

From a purely logical point of view, the obvious thing to do would be to change the overall design: instead of giving each gate-keeper all keys, *i.e.*, to give each driver the credentials to perform any hardware related operation, this should be restricted to the absolute minimum per driver/gate-keeper. This way, each gate-keeper can only cause problems in its own area but would not infect others. However, reality is not that simple: given the huge amount of work already put into the architecture and the software of nowadays computers, we simply cannot throw all of them away. Even then, the problem arises where to start trusting ones platform — since the only way to observe the operating system without significant effort is due to a program running on said operating system, one can not get any reliable information [9]. Hence, we need a solution which is able to co-exist with current hard- and software architecture.

In the picture of our university, we would fence one part of the ground, and partition it. Each partition does only allow *one* specific care-taker. If something goes wrong in this part, there can be only one person responsible for this. To allow communication between the care-takers in this new part, and also between care-takers of the old part and the new part, we install a special gate-house: each care-taker has a mailbox there, which only he can access. The gate-keeper's task is to make sure that each care-taker gets his and *only* his mail. This way, the care-takers can communicate, but nothing else. Similar, the outside world can ask the care-takers to perform certain tasks. In every single case, the care-taker in question can check if such a task would be allowed. Obviously, we now need to trust the gate-keeper, too. But given that there is only one gate-keeper with a very limited job, and that the hardware allows authentication of the gate-keeper, we can actually hope that this is possible. However, as we will see later, the role of the gate-keeper is quite critical when we evaluate the implications of the trusted computing technology. In particular, we need to know that the gate-keeper in question is actually real and not an imposter.

## 2 Overview Trusted Computing

After this rather pictorial introduction we move on to a more technical description of the gate-house, and the care-takers. We want to stress that the picture we drew on the previous pages actually is one of the most ambitious project in the area of trusted computing. Microsoft calls its internal project “NGSCB” (Next Generation Secure Computing Base, former Palladium), while the chip manufacturer Intel develops the underlying hardware under the name “LaGrande”. Its direct competitor AMD is working on a similar system called “Secure Execution Mode” (SEM). We will come back to these techniques later but want to stress that we may see something different in reality.

### 2.1 Trusted Computing Platform and Further

Without software support, a trusted PC will not behave different from a PC as we know it today. It will boot as normal, though a hardware module — the “Trusted Platform Module” (TPM) [10] will receive checksums of the boot sequence. However, the TPM itself is a passive module, hence some BIOS support is necessary to actually report the checksums. To this end, some part of the BIOS will be protected against manipulation and start reporting checksums to the TPM. In any case, this trusted platform module is a special chip, like a smartcard, which is bound to a specific platform — in most cases, this will be the mother board. The TPM chip will then save the current configuration, *i.e.*, the checksums of everything that in some sense affected the boot process, in a number of special registers. From a technical point of view, this is done by the mean of hash-chains, and the TPM chip will only save and later sign the last element of this chain. However, using cryptographically secure hash functions, this is actually sufficient and breaking this hash chain is far less likely than breaking the actual implementation of the TPM itself. With proper software reports, the TPM may receive another hash chain for the operating system and even another chain for the applications found on this computer. Any change in either the configuration will therefore lead to a completely different hash chain — which in turn makes it possible to detect that something has changed. Consequently, a manipulation due to a virus or a Trojan horse cannot go unnoticed anymore. Until here, we have the most likely scenario of trusted computing, as outlined in specification 1.2 of the trusted computing group (TCG), *i.e.*, the industry platform which brings all industrial players in this matter together.

Using LaGrande or SEM, the hardware goes one step further: now we are even able to separate the different drivers and programmes in a trusted and a non-trusted space. The operating system itself — plus all applications — are by definition untrusted: they are far too complex not to carry any security critical bug (see above). Moreover, they are usually not open source, so it is difficult for a user or even a big institution to establish any kind of trust that there is no trapdoor hidden. For all we know, Windows could be a big Trojan horse which is used to spy on all users. Even if this is not the case (we want to stress that the authors do not believe that Windows is a Trojan horse), we have no easy mean to rule this allegation out. Hence, trusting the whole operation system is simply very risky. And also for open source systems such as Linux, we have difficulties to establish the level of trust we actually need for our purpose: here, the problem is not open source, but the sheer complexity of the whole kernel, as well as a lack of people able and motivated to perform quality security audits on the code. Hence, it is simply not possible with nowadays means to ensure that the kernel does not carry any hidden weakness. Hence, we need to separate the code into a rather big, untrusted area and a much smaller, but trusted area, commonly called a “Trusted Computing Base (TCB) in the literature. There are a number of different implementations and names of a TCB that can support a legacy OS, such as the gate-keeper, micro-kernel, hyper-visor in the IBM model, or “Nexus” in the terminology of Microsoft. Its role has already been outlined above: deliver messages between the different care-takers, and make sure that they cannot interfere with each others tasks and in particular, that they do not steal each others secrets, *e.g.*, the PIN for a banking application.

### 2.2 Trusted Computed Platform Alone

While the advantage of the later is rather obvious, we will go back to the more short-term scenario of the trusted computing platform without any additional technology as LaGrande, SEM, or NGSCB and outline

how such a system could benefit users. In this context we assume a large company which has a big number of systems — both installed in the offices of their employees, and also for some of them at their homes. The threat this company faces are employees which install Trojan horses on the system without knowing, or bring in viruses into the internal network. In both cases, trusted computing may ensure that any change of the computer system will be noticed, and hence, the system administrators can inspect this machine and determine the cause of the change. This way, it becomes easier to maintain a big number of platforms and prevent the spreading of viruses or Trojan horses. If necessary, it is even possible to “quarantine” a platform which has been changed, and deny its access to the Internet or internal databases.

However, the trusted computing platform can more: using the technology outlined by the trusted computing group, it is possible to bind a file to a specific platform configuration and only allow it to be viewed on this platform. Thus — if the system is fully utilised, installing any programme, *e.g.*, a key logger, would render the file useless as the platform has changed. Note, however, that the TCG specification ends with verifying the master boot block — everything beyond that is up to the operating system vendor. Technically, this binding can be achieved using cryptographic techniques: each trusted computing module has a secret key which may be used to generate specific keys which only work in connection with a specific platform. If the configuration changes, the trusted computing module will simply refuse access to the key bound to this configuration and hence, the encrypted file can no longer be accessed.

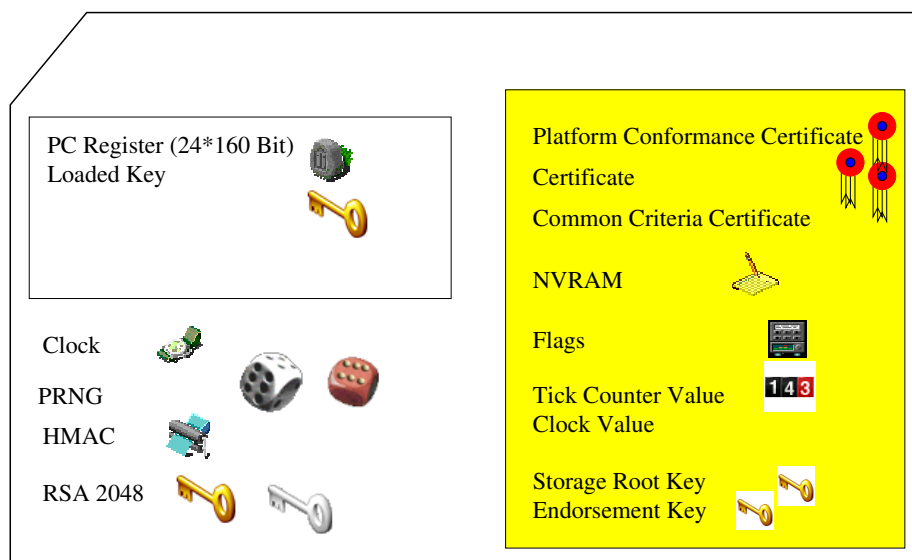


Fig. 1: Trusted Platform Module v1.2

As we see, the role of the trusted platform module is quite prominent here. As we need to start our chain of trust here, we actually need to trust the implementation of the TPM. Hence, each TPM comes with a private key which is unique to this TPM. Moreover, the manufacturer issues a certificate belonging to the public key of this private key, which basically states that the TPM has been manufactured in compliance to the rules laid out by the trusted computing group, and in particular, that the trusted platform module does not contain any hidden back-door. Moreover and to fulfil its duties, a TPM version 1.2 has the following hardware components:

1. Hardware real random number generator
2. Public and Private Key algorithms (RSA up to 2048 bit)
3. Hash algorithms SHA-1 and MD-5

The specification does not require support for any block encryption scheme, such as AES or 3DES. The reason is that in normal PCs, the TPM is a slow device with a low communication bandwidth, and that any ability to do block encryption would cause export regulations to apply on the technology. Hence, a question to ask may be if these export regulations are justified in the first place.

In this context, (1) is used to generate cryptographically secure keys, (2) makes sure that the private key never needs to leave the TPM and hence, it is impossible to impose as any given TPM, and (3) is used to compute the hash-chain for a given platform configuration, *i.e.*, for platform attestation. You can find an overview of a TPM in Figure 1.

In addition to these hardware components, we also have the following keys and certificates present:

1. Endorsement key (EK)
2. Storage Root Key (SRK)
3. Endorsement Certificate (EC)
4. Platform Certificate (PC)
5. Conformance-Certificate (CC)

The endorsement key uniquely identifies a given platform, and together with the endorsement certificate can be used to prove to the outside world that a given TPM is actually genuine, *i.e.*, does comply with the specifications. To guard the users privacy, it is possible to create pseudonyms by the mean of attestation identity keys, or AIKs. With these keys, a platform can prove to the outside world that it is a genuine platform but does not need to reveal its identity. As we will see below, there are some questions to be raised in this context, but at least, the TPM specification does take user anonymity into account. Finally, the storage root key is used to encrypt user keys. This way, a TPM can use the operating system to store its keys and thus manage a virtually unlimited amount of keys, even though its amount of internal storage is limited. User keys can have a number of attributes. Most importantly, a key can be marked as *non-migratable*, *i.e.*, it may not leave the TPM, and the key can be bound to a specific platform configuration, which makes it inaccessible if the configuration changes. However, an optional protocol exists to move such keys to another TPM, if it is guaranteed that the new TPM satisfies the same security requirements and the old key is deleted. This protocol is rather impractical to use, though, and has not been implemented so far. Finally, the platform certificate and the conformance certificate can be used to prove the correctness of given TPM as well as additional security features, *i.e.*, in particular that a given platform does not contain any known back-doors and can therefore be considered “safe” in a specific, application dependent sense.

### 3 Remote Attestation

The most critical aspect of the trusted computing concept — its ability to prove statements to the outside world — requires some further elaboration. This feature allows a platform to issue provable statements, for example about the platform configuration or the properties of a key. As the verifier of that proof has no reason to trust the platform or its user, the only entity he can trust is the TPM. Thus, to establish trust in the statements, he must be sure he is talking to a real TPM that properly follows the specification. Thus,

- a TPM must be able to prove it is a genuine TPM,
- if a TPM is hacked, it must be possible to revoke its key, but
- for privacy reasons the TPM must not be uniquely identifiable.

### 3.1 Original Implementation

The first implementation (TPM 1.1b) used a trusted third party (TTP) to establish anonymity. The user creates a pseudonym, *i.e.*, the Attestation Identity Key, and forwards it — along with the public part of the endorsement key and the certificates — to the trusted third party. The TTP verifies the certificates, and itself issues a certificate on the pseudonym. This certificate is then encrypted using the endorsement key, so that only the TPM can decrypt it. Thus, the TPM can get an arbitrary number of certificates that prove its genuineness, but is not linked to its identity as long as the TTP remains trustworthy.

The disadvantage of the scheme is that it requires a TTP that is trusted by both sides. This may be a trust problem, but also plain availability. For example, if TPM enabled devices want to set up an ad-hoc network, a commonly trusted TTP may not be available at that time.

### 3.2 Direct Anonymous Attestation

As a reaction to critics of the specification, the TCG has extended the specification with the Direct Anonymous Attestation protocol (DAA). This protocol is based on a zero-knowledge proof which allows the user to prove that he has a signature key, without ever revealing any identity related information. Keys can still be revoked if the secret key of the TPM is known to the verifier, *e.g.*, because it was published on the Internet, or if the same key is used with a very high frequency. While this protocol solves all above issues, it is rather complex, and it has to be seen if it will actually be used in practice.

### 3.3 Problems

Even though the issues related to the TPMs ability to prove its genuineness appear to have been resolved with version 1.2 of the specification, issues remain with the application of remote attestation. Some issues are due to the implementation. To demonstrate that an operating system satisfies certain properties, the verifier receives a log file of the entire boot sequence. This poses problems on both the privacy and the practicability side. For one, this may be more data than a user is willing to release — the BIOS version, for example, can be used to estimate the age and the price of the platform, and the operating system configuration can tell the users computer expertise. Also, it poses problems for the verifier — determining from a log file of a boot sequence if a platform satisfies a complex property may be a quite challenging task. Also, there is no transparency about the policy used. As the verifier gets all information about the platform he needs to derive his policy, he can also apply *any* policy, even those that may be considered illegal or unethical.

The second issue is that it is impossible for a user to lie about his platform configuration. Many national laws, for good reason, legalise lying in some situations. In Germany, for example, an employer is not allowed to ask a female job applicant if she is pregnant. As a refusal to answer is already giving away too much information, the applicant is allowed to lie without fearing consequences.

### 3.4 Owner Override and PCR Substitution

The concept of owner override and PCR substitution has independently been developed by Schoen [8] and Kursawe and Stueble [3]. The basic idea is to allow the owner of a TPM, hence usually the entity that paid for the platform, to modify the log-files of the boot sequence, *i.e.*, to lie about the platform configuration. While this abolishes all abuse scenarios, it also kills some of the good use-cases. There are perfect scenarios in which one does not want the owner of a TPM to be able to lie, for example, if the TPM is used to enforce a companies privacy policy. In other scenarios, it may be desirable to lie — for example, if a company tries to enforce a monopoly and lock the user to a platform configuration. For this approach to become feasible, some method needs to be developed that a human policy, *i.e.*, under which circumstances it is allowed to lie, can be enforced by a (rather simple) hardware module.

### 3.5 Property Based Attestation

Another approach, that addresses the above problems, is the concept of *property based attestation* [7, 5]. Instead of sending its log-files to the verifier, the platform receives certificates that its configuration satisfies certain properties, and binds them to the corresponding configuration. The verifier now only sees the certificates, but learns nothing beyond the fact that the platform satisfies the properties he asked for. This has several advantages. The users privacy is preserved, as the platform does not need to send out its log files. The verifier's task is rather easy, as he does not need to derive the properties himself anymore. And though lying is still not possible, this approach at least enforces transparency about the policies. The verifier must publish the properties he is interested in, which allows a judge to identify illegal requests and act accordingly.

In any case, we need to look into the question here, *where* these certificates actually come from and who pays for them — and under which authority the entity is which issues these certificates. For example, it would not help our cause if all big music distributors pool resources and set up such a certification authority. It would simply be too biased towards the needs of the music distributors in contrast to the users' and the artists' needs.

## 4 Application Examples

After introducing the technical specification of the trusted platform module and hence the most likely base for trusted computing in the coming years, we move on to a more case-based view, *i.e.*, we will now outline how a trusted platform module can be used in different scenarios and how not. As a general note, it should be mentioned that trusted hardware only serves as a basis for security applications, but does not do anything on its own. There are many ways it can be used to harden existing systems, but the real strength of the concept is to serve as a foundation for a secure operating system.

### 4.1 Company with computers in different places

We start by going back to the scenario already outline in Section 2.2 and assume a large company with computers installed in different countries and also both in its offices and at the home of some of its employees. Currently, the system administrators cannot be sure that the users really have a secure platform or — more likely — at least some of the platforms installed have a virus or a Trojan horse. The latter is a serious security risk when it comes to company secrets. With TPM 1.2, the system administrators can install the system once, and then “seal” it in its current state. Any modification of the system, *e.g.*, through a virus or a Trojan horse, could now be detected either locally by the user, or as soon as the platform accesses the company network. Moreover, the security policy can now make sure that no document can be read on a non-conforming platform. Data leaks become far less likely this way.

All in all, there are few ethical implications in this scenario and the benefits clearly outnumber the disadvantages.

While the TCG concept generally leaves the choice to the user, problems may arise if the user for some reason does need to access a certain service. This is the case if there already is a quasi monopoly, *e.g.*, chat programs, or video distribution, or if the service is only offered by a limited number of providers, *e.g.*, Windows Security updates or high bandwidth Internet access in remote areas. This puts the user into a sufficiently weak position that the service provider can dictate the policies; another potential scenario would be that a government defines policies that may not be entirely in the users interest. This, combined with the technological means to detect policy violations or even prevent them from happening in the first place, may pose a significant problem, though it can hardly be resolved by technical means. Hence, we need a critical public here which makes sure that these scenarios do not happen to begin with.

### 4.2 Secure mobile phone

Here we move from the world of personal computers to the domain of personal electronic devices, such as PDA or mobile phones. Particular for the latter, we may assume that the manufacturer has large control

over the programmes running on these items, though phones are slowly changing into open systems. Hence, it makes sense that this manufacturer certifies that his mobile complies with certain standards, *e.g.*, music downloaded to this mobile can only be listened to on this particular mobile and not somewhere else. The same goes for the distribution of games which may now be bound to this particular mobile and not played anywhere else.

From a business perspective, such mobiles are very desirable: these days, mobiles are omni-present, and with the additional property of being “secure”, they can be used to sell content securely; it is notable in this context that the same people that consider 1 Euro too much for a song are perfectly willing to pay the same price if the song comes in the form of a ring-tone. This again shows how interesting mobile phones are from a business perspective.

From a users perspective, using trusted computing on cellular phones has two opposing effects. For one, it may be more difficult to hack a phone — thus, binding it to a service provider who subsidised it or removing the noise a phone makes while taking a picture may become much more difficult. On the other side, phones will contain a technology for anonymous authentication, which will significantly increase the users control, assuming it is used properly.

### 4.3 Complex machine replacement parts

While technologically different, planes and cars have something in common: the need of replacement parts from time to time. Not surprisingly, there is a huge market for these items. A real problem in this context is fraud with replacement parts which do not conform to specifications. Imagine a plane crash due to such replacement parts. To protect their image, manufacturers have a real interest in making sure that these falsified parts can be identified. As it is state of the art to have more and more computer present in any of these devices, replacements could now carry a TPM unit which states that the replacement in question is actually genuine.

As long as cars do not talk to each other, there are few privacy issues with this. Matters become different as soon as they do — or more likely will in the foreseeable future. Moreover, parts not made by the original supplier of the machine are both: a security risk and a necessary part of open market competition. Hence, we need legal procedures in place which make it impossible for a manufacturer to “lock out” competitors. From a technical perspective it would actually make sense to have several authorities in place which are allowed to certify that a hardware item does comply with specifications and may therefore be safely used in a designated complex machine. Otherwise, we are likely to face serious industrial political questions.

In any case, the current specification of TPM 1.2 would actually allow such a scenario — both the desirable and the undesirable. From a pure technical point of view, there is few what we can do to prevent either of them but need laws in place to make sure we get more security without monopolies.

### 4.4 Digital Rights Management

Given the prominent role digital rights management played in the whole debate of trusted computing, we felt that we should include it in this article, too. However, we want to point out that TPM 1.2 itself can do only little in this context: it is rather unlikely to assume that all personal computers on this planet will be sealed to a specific configuration and not changed from then onwards. Moreover, it is unlikely that anybody would like to testify that such a complex machine does not contain any hidden weakness. It would simply be too costly to do all the necessary tests — compared with the revenue for digital music, books, or movies.

Matters become different if Microsoft’s NGSCB or any other secure operating system is in place. While the overall operating system and also the applications are not trusted by definition, this is not the case anymore for the Nexus and the secure agents, *i.e.* the gate-keeper and the care-taker from our above example. Here, we *would* have some trusted parts in which movies could be shown, music be played, or books be displayed. As soon as we are in this scenario, privacy issues become prominent: do we want companies to be able to track back the individual behaviour of a single user? While desirable from a company’s perspective, it is very questionable if it is from a society point of view. Hence, as soon as later version of trusted computing emerge, we have to watch very closely if they allow digital rights management and if so to which costs of

the privacy site. Again, a possible solution would be direct anonymous attestation as this makes tracing of users impossible.

## 5 Conclusions

Trusted computing will become a reality as there are simply too many players in the market who do want it. For closed user groups such as companies, we see clear benefits and do not assume ethical questions which could not be resolved with our existing legal and ethical framework, *e.g.*, workers' unions.

The main open issues we see are not in the area of technology, but in the legal framework surrounding this technology [2]. Given that we expect the trusted platform module being deployed on a global level with only national or (in the case of the EU) regional level legal frameworks, we can certainly expect some frictions here. The most pressing question can be expected in the near future about how to deal with monopolies of any kind. While some countries may accept these monopolies for the greater benefit of their own economy, other countries may reject them. Having TPM in place may force all governments to accept the same monopolies the same way, simply because of the power balance imposed. Unfortunately, we cannot offer a solution for this problem but state it.

A second question is actually technologically, namely how to modify the technology such that abuse scenarios become impossible, for example by allowing the users to lie about the status of their platform. By now, most proposals have serious implications also on desired use-cases, and thus the answer of the trusted computing group is negative. Currently, both researchers and civil rights groups such as the electronic frontier foundation actually are searching for a solution that would make both sides happy, but there is a risk that the search for the solution may take too long for it to be still practicable when it is found.

In the context of digital rights management — an often quoted application — there are still some open issues. For an highly dynamic platform such as Windows or Linux, the configuration of the system is simply too likely to change, and the operating system does not provide enough security for serious DRM to make sense in the first place. However, in an embedded platform like a digital video recorder, things are different. Here trusted computing as laid down in TPM 1.2 could actually be used. And as soon as LaGrande/SEM technology or NGSCB becomes available, things may be different on a PC platform as well. Still, some privacy and consumer protection issues remain open here and need to be addressed in any solution proposed.

All in all, the technology is very interesting as it will help us to get more secure computers. Actually, it is just the next step in a trend we have since years: the first personal computers with DOS did not have any door locked: all programmes in the computer could alter any content at will. The next step was the "protected mode" in the i286 generation which shielded applications against each other. However, on the level of kernels and drivers, there was not seen any need for such a shield. But we know better now. For actual security critical applications such as Internet banking or electronic signatures, we do need this shield and also trusted path from the keyboard over the processor to the screen. Given that this was already available in old mainframes, we are quite likely to get them soon in personal computer, too. The big question is for which political and social price, *e.g.*, in the context of anonymity and monopolies. Here, technical solutions are void. Instead, we need a social and legal framework in place to make sure the power balance keeps intact.

## References

- [1] Hewlett-Packard. *Trusted Computing Platforms: TCPA Technology in Context*. Prentice Hall PTR, 2003.
- [2] C. Koenig, A. Neumann, and T. Katzschmann. *Trusted Computing*. Recht und Wirtschaft, 2004.
- [3] K. Kursawe and C. Stüble. Improving End-user Security and Trustworthiness of TCG-Platforms. Technical report, Saarland University, 2003. <http://www-krypt.cs.uni-sb.de/download/papers/KurStu2003.pdf>.

- [4] Microsoft. Next Generation Secure Computing Base. 2003. <http://www.microsoft.com/resources/ngscb/default.aspx>.
- [5] J. Poritz, M. Schunter, E. V. Herreweghen, and M. Waidner. Property attestation—scalable and privacy-friendly security assessment of peer computers. Technical Report RZ3548, IBM, 2004.
- [6] H. Reimer. Schwerpunkt: Trusted computing. *Datenschutz und Datensicherheit (DUD)*, (9), 2004.
- [7] A.-R. Sadeghi and C. Stübli. Property-based attestation for computing platforms: Caring about policies, not mechanisms. In *New Security Paradigms Workshop*, 2004.
- [8] S. Schoen. EFF comments on TCG design, implementation and usage principles, 2004. [www.eff.org/Infrastructure/trusted/\\_computing/20041004\\_eff\\_comments\\_%tcb\\_principles.pdf](http://www.eff.org/Infrastructure/trusted/_computing/20041004_eff_comments_%tcb_principles.pdf).
- [9] K. Thompson. Reflections on Trusting Trust. *Communications of the ACM*, 27(8):761–763, 1995.
- [10] Trusted Computing Group. TPM Specification version 1.2. 2004. <https://www.trustedcomputinggroup.org/>.