

Chapter 2

Futures for Trusted Computing

Abstract Trusted virtualisation is anticipated to become the dominant form of Trusted Computing in PCs and servers because it enables isolation of applications, and simplifies determination of platforms' trust and security properties. Trusted Computing can enable platforms to provide trusted services such as cryptographic erasure of data, negotiations for the supply of services, single-sign-on, and digital signatures. These provide greater confidence in the use of computer platforms. Nothing is free, however, and Trusted Computing is no exception: it requires a public key infrastructure and other infrastructure that is peculiar to Trusted Computing.

This chapter extrapolates existing technologies and trends. It speculates that trusted virtualisation will become the dominant form of Trusted Computing (in PCs, at any rate), describes some potential usages of Trusted Computing, and describes some of the infrastructure that is necessary to make it happen.

2.1 Trusted Virtualisation

It is anticipated that future computers will use trusted virtualisation, to prevent applications attacking other applications. This is because the only known generic way of preventing attacks by software on software is software isolation. If software can't touch data and the applications that use that data, the software can't misuse the data or subvert the applications.

Future computers will use hypervisors to provide separate OS environments, possibly enhancing separation via execution on separate physical processor cores. Selected data and applications will execute in separate OS environments, so they aren't affected by what is going on in other OS environments. The hypervisor will control the creation and destruction of the OS environments, and control communications between environments and with other platforms. Trusted platform technology will be used to ensure that secrets belonging to a particular hypervisor are only revealed to that hypervisor. In some trusted computers, trusted platform technologies will release keys to hypervisors executing in the isolated environment provided by new platform architectures. Other trusted computers will comprise a

hypervisor that controls the platform's hardware and provides trust functions to various virtual computers supported by the hypervisor (perhaps executing on different hardware cores), and control data flows to and between those virtual computers. For more details see Chap. 13.

One trusted piece of firmware is the first software to execute when one of these trusted virtualisation platforms boots. This firmware measures the next software to execute, stores that measurement in the TPM, and then passes control to that software. The software does its normal work, measures the next software to execute, stores that measurement in the TPM, and passes control to that software. And so on. Eventually the hypervisor is booted. Remaining software (apart from upgrades to the hypervisor) will be executed at a lower hardware privilege than the hypervisor and therefore can't subvert the hypervisor. At this point it becomes unnecessary to record new software (apart from hypervisor upgrades) in the TPM, because the TPM already contains measurements of all the software that can affect the integrity of the hypervisor.

Once the hypervisor has booted, it accesses the hard disk drive and retrieves encrypted files. The hypervisor then asks the TPM for the keys to those files. The TPM compares the software measurements stored with the keys against the current measurements. If they are the same, the hypervisor is the legitimate owner of those keys, and can be permitted to use the keys to decrypt the files. Otherwise the requestor is not the proper owner of those keys, the TPM will refuse to use or release the keys, and the encrypted files cannot be decrypted. Hence a platform could boot using rogue software, but that software won't be able to access the user's data, won't be able to display protected images, and won't be able to use a TPM's signing keys to fool other devices. This uses a mechanism unique to Trusted Computing that is called "sealing", which restricts the availability of small amounts of data and keys. Sealed data is encrypted data that contains both plaintext data and plaintext measurements of the software environment that must exist outside a TPM before the plaintext data can be used by that software environment. When sealed data is created, a TPM concatenates plaintext data and plaintext measurements of a software environment, and encrypts the concatenated data. When sealed data is decrypted inside a TPM, the resultant plaintext measurements are compared with measurements of the current software environment. If the two measurements don't match, the TPM refuses to allow the current software environment to use the plaintext data from the sealed data.

Computer users should see very little change when they run applications on these platforms (and deliberately so). They will, however, need some means to determine whether the computer in front of them is in a trustworthy state. The fundamental obstacle is (obviously) that a device can't make a decision about its own trustworthiness. There's no way it or a user can believe a platform's own assessment of itself. Instead, the decision must be made using another computing device, either directly attached to the computer (via USB, for example) or indirectly connected via a network (for example). That other device must ask the target computer for the measurements that it made during boot, interpret them (to decide whether the platform is executing a respectable hypervisor) and then

display the conclusion to the user. Once the user believes that a computer is in a trusted state, the user can introduce personalised images (such as text, shapes and pictures) to a trusted function in the target computer. The target can then use those images as the background or circumference of trusted windows on a normal display. When the user sees a window that uses the personalised image, he knows that that window was generated by trusted processes.

Note, however, that a crucial step is missing from the previous description. It is meaningless to ask a platform for its measurements unless one knows that the platform will respond truthfully. Trusted platforms therefore need a way to prove that they are trustworthy, and will provide genuine measurements. At first sight all that is necessary is to install a single cryptographic signing key in a genuine platform, provide a certificate to attest that that key belongs to a genuine trusted platform, and ask the platform to sign measurements with that key. This works perfectly but has the side effect that any interactions with that platform can be correlated. A third party can tell that the same platform was used to file a tax return, shop for vegetables, and read the news, for example. The correlation is indisputable because a signature from a strong cryptographic key can't be spoofed. This is one reason why some commentators believed that trusted platform technology would degrade privacy. In fact, privacy is impossible without data protection (whether provided by Trusted Computing or other security mechanisms), because anyone can snoop on data if it isn't well protected.

The real issue is that “audit is incompatible with anonymity”, because good security requires a good audit trail to enable investigation of attacks or breaches of policy, while good anonymity requires a weak audit trail, to prevent investigation of previous events. Modern zero-knowledge-proof security techniques can actually improve privacy because they can prove possession of specific attributes, privileges and properties without revealing identity. (There's no longer any need to identify someone in order to check that they have the right to do something.) Hence Trusted Computing's design objective is to provide a good level of security and optional anonymity. In some situations it is possible to be anonymous and in others it is not. One needs a good audit trail for bank records or medical files but wants anonymity for more trivial actions such as browsing web sites, for example. TCG therefore provided ways for genuine trusted platforms to obtain any quantity of separate signing keys, called Attestation Identity Keys (AIKs) in TPMv1.2 (Attestation Keys in TPM2.0), any of which can sign and produce evidence that measurements are genuine. An Owner can then use one AIK when filing a tax return, another to buy carrots, and another to read the news, and so on. And, of course, the Owner can still choose to use one key for multiple activities, if he wishes.

One way of getting AIKs involves a third party called a Privacy Certification Authority or Attestation Certification Authority, and can be used to produce correlated or uncorrelated AIKs, as desired. Another way uses a zero-knowledge-proof technique called Direct Anonymous Attestation, which enables a TPM to directly setup an AIK with an interested party. These AIKs can be anonymous (meaning that it is computationally infeasible to tell whether the same TPM

generated different AIKs) or pseudonymous (meaning that it is possible to tell that the same TPM generated different AIKs, but not which TPM).

TPMs contain several other classes of functionality intended to be used by the hypervisor, but it remains to be seen whether hypervisors will actually use those functions. If not, we anticipate that those functions will eventually be deprecated and ultimately deleted from the TPM.

- Some functions have a high probability of being used. These include delegation of privilege (where the hypervisor performs management tasks that are normally the privilege of the platform's Owner) and monotonic counters (which the hypervisor uses to detect attacks using genuine-but-old versions of itself).
- The future of some functions is in doubt. These functions include the ability to create an audit trail of TPM usage and the ability to do time stamping. The issue with auditing is that the TPM's audit trail is no use without a hypervisor to maintain a log, so why not just use the hypervisor to create an audit trail?
- The future of some functions is difficult to predict. These include DAA (Direct Anonymous Attestation), which seems to provide little advantage in the enterprise environment because enterprises have no need for pseudonymity for in-house computing.

2.1.1 Privacy Implications of Trusted Virtualisation

Trusting a trusted platform requires the exchange of measurements. If a platform has common-place software, measurements don't disclose much about a platform—it's merely one of many platforms that have that environment. On the other hand, if a platform has specialist software, some commentators fear that revealing measurements is sufficient to uniquely identify that platform. This is actually less of a problem than it first appears, since a legitimate challenger doesn't require a detailed breakdown of the software on a target platform. A challenger really only needs to know what hypervisor can be supported by a platform. Then the challenger (perhaps after negotiations with the target platform) requests the platform to load a particular OS and set up a particular topology of particular protected processes. The challenger should have no legitimate interest in other processes in the platform, since the hypervisor provides application separation, and other processes should be irrelevant if the hypervisor provides sufficient process separation. It is of course true that an optimised hypervisor might serve to identify a particular platform, but even this is less likely than might first be thought, because a hypervisor must be one that is recognisable (and trustable) by a challenger. Otherwise the challenger has no way of deciding whether to trust that hypervisor. It follows that any hypervisor used for trusted operations in a public context is almost certain to be a well-known (and widely known) hypervisor, and hence unlikely to be a one-off creation that identifies a particular platform. (A hypervisor in a private

context could be both a one-off and trusted, but presumably it doesn't matter that the platform could be identified, because of the private context.)

There might be ways to ameliorate risk when revealing measurements, but these are not currently part of any Trusted Computing specification:

1. One way is to use zero-knowledge-style protocols to prove that a target has some desired properties without actually revealing a large set of platform properties. Unfortunately no one really knows how to do this.
2. Another way is for measurements to be interpreted by a third party, which effectively blinds the measurements and provides a simple (perhaps) Boolean answer to the question "does this hypervisor meet my requirements?". The problem here is that there may not be a business case for such third parties, and there may be doubts about their trustworthiness. (This is the same argument used to justify the use of the Direct Anonymous Attestation protocol instead of using Privacy-CAs.)
3. Yet another way is to use the isolated execution environments (compartments) provided by the challenger or the target. A compartment can host a policy interpreter that examines measurements to verify conformance with policy, while hiding the integrity metrics from other processes in a platform.

Of course, with options (1) and (3), a platform still needs to expose sufficient integrity metrics to prove that it can provide the necessary functionality. Even this can be avoided, however, if that functionality is an inherent part of a platform, and proof of attestation identity is sufficient proof that that functionality exists in a platform. Then potentially no measurements need be disclosed by the platform. A "policy checker" that is an inherent part of a platform should be considered a type of "Root of Trust" of that platform. Otherwise at least the policy checker must be measured and the actual measurements reported to the challenger.

We note in passing that method (3) could also be used to alleviate conventional intrusion attacks. It is common for platforms to be scanned, looking for weaknesses. But if all connection setup were negotiated within a compartment, connection could be refused unless the process in the compartment was a known "good-faith" connection program. Both the enquirer and the target would use trusted platform technology to verify that details from the target would be interpreted only by a known "good-faith" connection program. The enquirer would have access to only the sanitised output of the compartment, so the target should have no qualms about misuse of detailed target information.

2.1.2 Virtualised Trusted Platforms

In computing, the term "virtual" implies non-hardware interfaces that mimic hardware interfaces. Software that executes on a virtual platform can be the same as software that executes on a hardware platform. The difference is that software executing on a virtual platform executes on an interface that mimics a

hardware interface, instead of on an actual hardware interface. In the case of true virtualisation, the software is completely ignorant of the fact that it is executing in a virtual environment. In the case of para-virtualisation, the software is optimised to take advantage of the virtual environment, or to allow optimisation of the virtual interface.

Virtualised trusted platforms are platforms where the software components of a normal trusted platform execute on an interface that mimics a hardware interface, in separated environments in host trusted platforms. In individual virtualised trusted platforms (such as PCs and servers, but probably not mobile phones), the evidence that the platform will properly protect keys and report measurements is provided via a certificate (or certificates) that vouches for a secret signing key held by a TPM. The question is what evidence is required for a trusted platform executing on a virtual hardware interface? There are two possible answers.

- The first possibility is that the host platform providing the virtual hardware interface provides the certificates, and the host platform provides evidence that it is trustworthy enough to provide the certificates. This is done via measurements of the host's boot process and via its certificates.
- The second possibility is that the host's manufacturer provides certificates for the Virtualised Trusted Virtualisation Platform.

Whatever the case, whenever the virtual platform provides its measurements, it must supply its own measurements and the evidence that the measurements can be trusted.

2.2 Future Trusted Services

This section speculates on services that might be provided by trusted platforms. There are many unknowns.

2.2.1 *Data Deletion*

Trusted Computing enables a form of data deletion, via the reliable destruction of encryption keys. The keys can be protected by TPMs, or can be inside self-encrypting-drives. Once all keys are erased, the data can't be accessed even though it still exists.

Key erasure is a two-edged sword, of course. Once a key is really gone, no amount of wailing or gnashing of teeth will bring it back. Computer users should therefore be equally concerned (and arguably more concerned) about reliable and continued access to data. This is why Trusted Computing provides means to control the distribution and duplication of encryption keys.

2.2.2 *Contracts and Negotiations*

Given platforms with isolated execution environments, any arbitrary service can be described in terms of a set of processes and each process can be allocated to a particular environment. Each process continues to be described in terms of its properties, resources and quality of service (as normal), but isolation introduces new attributes: (1) the controls that must be enforced on input data; (2) the controls that must be enforced while executing data; (3) the controls that must be enforced on output data; (4) the controls that must be enforced on audit data. Extra steps are needed in the process of negotiating contracts for services, and methods for executing those services on a computing platform according to the contract.

Contracts include a specification of the methods and processes used to perform a service. The contract specification should be capable of interpretation by a computing platform, and partition the methods and processes into functions that must be trusted if the service is to be trusted, functions that are merely required (as a matter of choice) to be trustworthy, and functions that have to operate properly for the service to function, but don't need to be trusted and/or are not required to be trustworthy.

Functions that must be trusted include those provided by a trusted platform that are used to report on the state of the software platform in a computing platform. Functions that must be trusted also include methods that will provide evidence of the execution of the service. Functions that are required to be trustworthy are functions whose integrity is paramount in the opinion of at least one of the parties involved in the contract.

Negotiations may require new service types (in the computing sense, where "type" indicates the operations that can legitimately be performed on data). The input data to a service and output data from a service would be typed, and a process would be typed. Typing could state the quality of isolation that is required, whether services may swap between isolation environments, and scheduling of swaps. Platforms could contain a label indicating the presence or potential presence of a predetermined software state. Labels should be global, in the sense that the same label always describes essentially the same services, and are signed by some trusted entity.

Receiving a challenge that contains a label should cause the receiving platform to determine whether it can provide the service described by that label. Platforms could publish the labels of all the services they can support, even if not currently providing those services, and use signed labels to decide whether to use another platform for a particular service. Labels could be differentiated to indicate facilities optimised for client-side or server-side operation, descriptions and certifications of labels could be broadcast throughout a domain, and platforms could use web services to advertise that they may support a particular service.

Audit parameters need to be specified. These include:

- The format of logs that record integrity metrics plus the method of their measurement

- The quantity of hash engines used to create logs
- The frequency with which input data and/or output data and/or program instructions are sampled.

2.2.3 Single Sign-On

The concept of single sign-on is well-known: a platform authenticates a user and then automatically represents the user to networks using various (and differing) authentication techniques and tokens. Trusted platforms improve on the concept, because they can use attestation identities and measurements to prove to the network that user authentication was properly done and that any particular network authentication method was executed as expected.

Trusted platforms can, however, extend the concept. If a platform is trusted by a network, any user authentication normally done by the network may instead be done by the trusted platform on behalf of the network. Then, whenever a network receives a connection attempt by a known trusted platform, the network simply grants access to the platform, knowing that the platform will already have authenticated the user on behalf of the network. (The network accepts access requests based on a user name and the platform's identity, knowing that the platform has already authenticated the user and verified his privileges.) This potentially minimises the number of network secrets (or private data) used as authorisation data, and hence reduces the complexity of maintaining the PKI within an organisation. The number of secrets in a domain is reduced to a minimum, yet individual users may still be identified and access to applications may still be individually controlled. At the same time, domain security may be maintained across the domain by a broadcast mechanism, without having to deal with each platform as an individual.

The end effect is that each trusted platform verifies the network on behalf of the user and verifies the user on behalf of the network.

2.2.4 Trusted Software Agents

Software agents would be much more useful if they could execute on trusted platforms, because they could carry private information (including encryption keys) with them, knowing that the confidentiality of that information would be respected. The problem to be solved is the propagation of private data through a platform or a network, depending on the trust properties of the destination. Data should be accompanied by policies that dictate the permitted usage of the data, including the extent to which it can be forwarded. Data could be accompanied by dummy data (for testing) and a "release public key", used to encrypt the work done by an agent and release its data into the wild (so that its legitimate owner might have a chance to recover it) when a platform can no longer be trusted to process real data.

Preferably platforms would have keys that can be irrevocably erased, so that future access to encrypted agent data would be irrevocably denied.

If trusted systems are the norm, and the confidentiality of information is respected by platforms, private information (information distributed under tight control) such as credit card details can be used for authorisation purposes instead of secret information, to request access to a service. The requirement for a domain's on-line public key infrastructure (PKI) is reduced, or even eliminated. Continuing the label theme introduced earlier, domains may not even be aware of the name of a caller, simply the name and label of the calling platform.

2.2.5 What You See Is What You Sign

Just because a platform is a trusted platform, it doesn't mean that the platform is safe to use. A person using one must be able to tell that it is operating properly before doing sensitive tasks, such as digital signatures. Digital signatures may be legally binding, so a person should always have checked that his trusted platform is in the correct state before using it for critical tasks.

Although trusted platforms are inherently designed to enable a third party to deduce the current state of the software platform, these techniques involve complex processes that cannot be done by an unaided human. A person wishing to use these techniques must therefore interrogate a platform using a separate computing device, such as another computer, a USB device, or even a mobile phone (for example).

Another option is to build trusted platforms and software platforms to display an image around the edge of a window on the monitor, or as the background to that window, for example. If an encrypted version of a personal image is locked to a desired software platform inside a particular trusted platform, a person who sees that image could have confidence that the platform is in the desired software state.

2.3 Infrastructure Requirements

Trusted platforms create new infrastructure requirements for manufacture, installation, maintenance, and logging.

2.3.1 Public Key Infrastructure

The most basic infrastructure requirement is that of a public key infrastructure. All trust in trusted platforms comes from people and organisations (including commercial companies). Trust is expressed via credentials that attest to genuine trusted platforms, genuine values of software measurements and genuine Attestation

(Identity) Keys. At the very least, it is essential to be able to verify the signature on a credential. In all but the simplest of situations, this requires certificates, each stating the public key and name of the entity whose signatures can be verified using that public key. The certificates can be arranged in a hierarchy, called a public key infrastructure, where the public key in a parent certificate can be used to verify the signatures on child certificates. Generally the root certificate is signed by a respected and well-known entity whose public key has been well publicised by other means. The root certificate can be verified using that root public key, and hence all certificates in the hierarchy can be verified.

Some Trusted Computing Group infrastructure specifications describe a schema for the evidence of trustworthiness of trusted platforms. In particular, the “TCG Credential Profiles” specification describes the credentials used to support a trusted platform. Other TCG specifications allude to (but do not specify) the actions of an entity called a Privacy-CA that signs credentials for Attestation Identity Keys.

Few of TCG’s infrastructure specifications are implemented. It’s a chicken-and-egg problem. TCG’s infrastructure specifications are not implemented because they communicate attestation, which requires features of X.509 certificates that are not implemented in common software libraries because no one is using attestation. For example, a platform certificate is fundamentally an attribute certificate, because it contains no key. There’s no sign that this deadlock will break any time soon.

2.3.2 *Manufacture*

Although this aspect is invisible to customers, it’s instructive to note that Trusted Computing makes new demands on manufacturers.

Manufacturers are unique in the Trusted Computing life cycle (from product creation to product destruction) because products do not have to conform to trusted platform specifications whilst they are in the hands of manufacturers. The reason is (obviously) that a product isn’t expected to satisfy a specification before it is finished. The effect is that a product doesn’t have to obey any specifications until the manufacturer says that it is finished, but (on the other hand) the manufacturing process must have certain properties in order that the finished equipment can satisfy those specifications. The general requirement is that the manufacturer must ensure that any product it certifies is actually worthy of certification (does actually satisfy the specifications). Otherwise the manufacturer’s “word” is worthless and its brand reputation is damaged.

This aspect of production is new for most mass-market manufacturers. While their products may be supplied with warranties, it is rare that manufacturers attest that individual specific pieces of equipment were manufactured by them to meet specific specifications. Thus manufacturers must modify their production techniques and install the ability to sign credentials. (It isn’t always true, of course, that all manufacturers need modified production lines. The primary exceptions are

those TPM manufacturers who also manufacture smartcards, because they typically already have processes in place to certify individual products.)

2.3.3 Upgrading TPMs

TCG specifications permit TPMs to be remotely upgraded, although it is not mandatory. Field upgrades are intended to permit correction of problems or installation of improvements, such as more modern versions of the TPM specifications. Any field upgrade requires permission from both the manufacturer (probably via a signature) and the current platform Owner (either cryptographically or via direct physical interaction with the platform). The entity providing an upgrade must ensure that the upgrade does not permanently prevent access to any data already protected by the TPM, or expose any data already protected by the TPM.

2.3.4 Upgrading Integrity Metrics

Trusted Computing relies upon good change-control practices, to ensure that software with attested software measurements has good behaviour, and to ensure that the proper interpretation of measurement values is well-known.

Trusted platforms use measurements of software programs as an indication of platform behaviour. Measurements are used for three purposes, and changes to measured values can cause denial of service.

- Measurements are signed using attestation keys and reported to challengers. If measurements change, the challenger must interpret the new measurements in order to decide whether the new platform state is acceptable.
- Data is sealed with measurements, and a TPM will not reveal data if the sealed measurements do not match current measurements. Data is sealed in order that only specific software environments can recover specific plaintext data. Environments can do whatever they wish with that plaintext data, including (in principle) seal it to different measurements. Hence, if all environments include a re-sealing facility, it is unnecessary for TCG to specify a method of data recovery after measurements change. However, providing a re-sealing facility in every environment is onerous.
- Migratory and duplicable keys are used to backup and transfer protected data between environments. They can be sealed to measurements, just like data. TPMv1.2 in particular doesn't provide any method to enable keys to be sealed to new measurements, whether in the original platform (because software has changed) or in a target platform (because the new platform has different software).

Measurements are deliberately designed to change whenever a program changes by a single bit, because even tiny changes in a program might cause undesirable changes in behaviour. However, this means that measurements of software change even if the trustworthiness of the software has not changed. The problems caused by this “brittleness” can be severe. Firmware revisions aside, even platforms with the same model numbers may have different chipsets due to configuration options and commercial decisions (such as alternative vendors and dual sourcing). Thus even a constrained set of hardware platforms may have many trust-equivalent software measurements.

There is no way to ameliorate brittleness in measurements in TPMv1.2, but TPM2.0 has a feature called Enhanced Authorization that allows in situ adjustments of measurements attached (sealed) to keys. Further, TCG’s “Dynamic-RTM” specification allows measurements to be expressed using a public key. Using a public key transforms recorded and reported measurements into measures of what something does, instead of what it is. An actual software measurement is verified using a policy (a public key), and the name of the policy (the public key) plus the outcome of the test are recorded and reported as the measurement. Then, when firmware or software changes, the actual measurement changes but the policy and the outcome of the test do not (assuming the new software is as trustworthy as the old software). Thus the recorded and reported measurements do not change.

An even better method might eventually be derived from the semantic web, by translating behaviours directly into measurements.

2.3.5 Auditing Trusted Platforms

Secure platforms produce an audit trail, and trusted platforms should do the same, to enable investigations after bad things have happened.

Audit data is evidence that a process executed, and how it executed. Audit data is typically used to resolve arguments over “who did what”, and “how”, and “how well”. The data might simply record aspects of a process, or may enable that process to be totally recreated and reviewed. An engine gathering audit data would normally observe all data entering an isolated environment (including the process software itself), all data leaving the isolated environment, and the execution of the process (instructions executed and memory altered). Recording the actual execution of a process would tend to produce a large amount of data, so it may be preferable to sample the execution process. This could be done by sampling at regular or irregular intervals, according to a keystream generated from a particular key or secret.

Preferably, for privacy, there should be different views of audit data: one view reveals all input data, one view reveals all data in the vicinity of a marker during execution, one view reveals all output data sent to a particular destination, and so on. This provides privacy for audit data, in the sense that an enquirer may be offered access to certain aspects of an audit trail but denied access to others. Audit sets

would be useful particularly when investigating processes that fail to execute as required. A first set of audit data may provide access to all audit data for a specified time period, for example. A second set may provide access to a subset of the audit data for the duration of a given process, for example. A third set may provide access to a subset of the audit data in the presence of certain events, for example. One set may provide access to all attributes of all data in some time period, while another set may provide access to selected attributes of one data parameter for the entire process, for example.

Policies may determine what audit sets must be created and when can they be revealed. The default would be to contact the owner of the data being processed.

2.3.6 Discovering Trusted Services

This section concerns the discovery of trusted platforms, and how to decide whether to trust them. We preferably need a method that works with existing computer systems because it will be years until all installed computers are trusted platforms. Until then, there will be a mixture of conventional computers, first generation trusted platforms, and trusted platforms.

The first generation of trusted platforms provide a low-cost hardware-based crypto API to a TPM for existing applications to store cryptographic keys. The next generation of computers will have hypervisors that additionally provide measurements to enable new applications to identify software platforms, and provide controlled (isolated) execution environments (software compartments). Such platforms can map services in terms of a network of processes, each with its own isolated allocated internal resources, permitted connections, and forbidden connections. The trusted platforms hosting these isolated processes can provide certificates and measurements as evidence of the controls and mechanisms protecting processes, and evidence of processes being executed, their resources and connections. The trusted platform knows nothing about what the processes are doing, merely that they are connected together in a particular topology that is both reported and enforced. If services and policies can't be mapped this way, trusted platforms cannot properly protect them. It follows that all services and all policies should be described in terms of a network of measured protected processes. There seems no fundamental reason why any service or policy could not be described in such terms, but it is not always easy to "get there from here". It is currently unclear whether practical policies can be expressed in canonical terms.

Given that trusted platforms enforce networks of isolated processes, we suggest that discovery and setup of services on trusted platforms falls into six categories:

1. Evidence that the other five categories of information can be believed.
2. Evidence of mechanisms to instantiate black-box execution environments (separate threads and/or processes).
3. Evidence of controls that govern the inputs to those black-box environments.

4. Evidence of controls that govern execution within those black-box environments.
5. Evidence of controls that govern the results from those black-box environments.
6. Evidence of controls that govern the audit trail of those black-box environments.

This partitioning simplifies the interpretation of trusted platforms and makes it simpler to decide how to allocate platforms to particular tasks. The latter five stages are compatible with existing ordinary platforms (that don't use trusted platform technology), and hence work with systems that contain both ordinary and trusted platforms.

The level of protection afforded to data in a platform naturally depends on the environment implemented by the platform. Some platforms provide lower levels of separation of execution than other platforms. For example, conventional common operating systems provide separate threads or processes, but enforcement of separation may be weak and is typically under the control of the platform's administrator, who can override. Less common types of existing platform might provide stronger separation using software or hardware compartments, and the administrative role may be partitioned, so collusion by multiple rogue administrators is required to subvert separation. Second generation trusted platforms should provide strong levels of separation and prevent administrators from subverting that separation.

According to this proposal, in stage 1 a challenger checks the (attestation) identity of a platform, to discover whether it is a trusted platform or an ordinary platform. If the target is an ordinary platform, the level of trust in the platform is uncertain (although the platform may, of course, be perfectly trustworthy). It may, however, be difficult to be sure that the target platform is even the platform it claims to be. If the target is a trusted platform, its attestation identity identifies the platform, and measurement data signed by that identity can be relied upon.

The challenger then uses stage-2 to understand the type and degree of process isolation that can be provided by the target platform. It may be that the target environment uses just standard conventional OS technologies to separate processes. (If the target is a first generation trusted platform, the attestation identity indicates that the target is suitable to support a conventional OS and integrity metrics signed by the attestation identity show that the OS was properly booted.) It may be that the target environment is a specialist cryptographic environment, in which case an attestation identity vouches that that environment exists. Another possibility is that the target is a trusted platform where one OS is isolated from another by virtue of using hardware processor cores, or where a hypervisor separates OSs using software. In these cases, the attestation identity provides evidence that the target is suitable to support the environment; integrity metrics show that an appropriate kernel/hypervisor was properly booted and the kernel/hypervisor provides evidence of isolation of applications or other OSs.

Data from stage 2 should also reveal additional properties of target platforms. It may be that some platforms can provide protected human interfaces, in the sense of keyboards that cannot be snooped or subverted, or displays that are guaranteed to

accurately present the visual output of a process. Other hardened platforms may have disk drives and optical drives with security properties, for example.

Once a challenger receives stage 2 data, he can deduce the characteristics of the target and decide what processes (if any) could be executed on that type of platform. Any arbitrary task can be partitioned into (parallel and/or serial) processes that must be (or are desired to be) protected to some degree. The challenger should partition his task appropriately and determine which target is suitable for a particular job. The challenger should do this by assessing the sensitivity of data that passes between processes. Some data might be insensitive, in which case it does not matter whether it can be interpreted by any arbitrary process on the target that produces or consumes the data. Other data might be sensitive, and capable of interpretation just by desired processes, but no others. A platform could enforce such restrictions by controlling the flow of data between processes. Otherwise there seems little alternative to encrypting data, and ensuring that only intended processes have the keys to encrypt and/or decrypt messages between processes.

The challenger should also decide what processes can observe the actual execution of a job (view the execution of instructions and/or the memory locations used by the process). It could be that no additional process should be able to inspect the execution of a process, or that only an administrator-privilege monitor is permitted to observe execution, or only in the vicinity of execution faults. The challenger should also decide whether an audit trail is required, and (if so) who should have access to it. Note that we use the term “audit trail” in its most general sense, including the ability to reproduce or recall the execution of a process. The audit trail could be created by sampling the audited process, rather than be a complete record of the process.

Once the challenger has made all these decisions, he should request each target to customise its environment to provide the desired levels of protection for input, execution, output, and audit. The target platforms make any necessary changes and provide the challenger with measurements that are evidence that mechanisms are in place to enforce the desired policies. After receiving and verifying this evidence, the challenger distributes his task amongst the targets, and work commences.

One potential stumbling block is the infrastructure required to support public Trusted Computing. While a public key infrastructure can work perfectly well within a closed organisation, many commentators have expressed concerns about scalability. In the public domain, some sort of infrastructure is necessary to enable verification of the various certificates that vouch for the various parts of a trusted platform, and the platform itself. It may not always be possible to reach and traverse a database to validate primary trust sources, and hence secondary sources may be desirable. We therefore propose the installation of distinctive kiosks in selected locations, to provide introductions to trusted platforms in the locality. Thus an automated information kiosk in the reception of a building could contain credentials of trusted platforms in that building, or export a public key that can be used to validate credentials of trusted platforms in the building. The presence of the kiosk in such a prominent place would be taken as evidence that the kiosk could be trusted. Naturally, the kiosk must be protected from unauthorised manipulation, and

hence may itself be a trusted platform. Alternatively, a mobile ‘phone could be used via a network to authenticate a person to a platform, prove to the person that the platform is trustworthy, and cause their data to be delivered to that platform, so the platform can be used by the person to operate on their data.

Trusted platforms distinguish themselves via their attestation identities and measurements: attestation identities and measurements are evidence of whether a platform can support a given service. We suggest that sets of possible measurements should be advertised by platforms whether or not they are currently in the state indicated by those measurements, and when a platform receives a request indicating one of its attestation identities and an appropriate set of measurements, the platform instantiates the appropriate state “on demand”. In this case, the measurements should include a set of applications that the platform is able to instantiate. Then a service-requestor can simultaneously indicate a request for a particular service executing on a desired OS, and the measurements become (in effect) part of the service’s identification. A requestor looking for a particular service could interrogate a database or uses a search engine, looking for a service identified in terms of a set of measurements with the intent of collaborating with existing providers to use the service or to augment the service.

Customers will require some indication of a service’s trust status. Native measurements are unintelligible for ordinary customers, so the process must be automated and customers must have a simple way to initiate automated checking and view the results. This could be achieved by appropriate user interface design: “right clicking” on an icon representing a platform or service could present the option of challenging that platform or service, and the icon or service should change according to whether its trust status is unknown, acceptable, or unacceptable. The source code of a program could change colour depending on whether it is executing in a platform whose trust status is unknown, acceptable, or unacceptable. Dragging a platform icon onto a service icon could cause the platform to test whether that platform can provide the service.

Trusted Computing Platforms

TPM2.0 in Context

Prouder, G.; Chen, L.; Dalton, C.

2014, XVIII, 382 p. 9 illus., 2 illus. in color., Hardcover

ISBN: 978-3-319-08743-6