
**ECE750-6: Pervasive Computing**

**Final Project**

# A PKI for Trusted Pervasive Computing

Michael Jarrett(99318764)

August 28, 2004

# Abstract

Trusted computing is a new research area attempting to provide additional security to computer networks by allowing trusted software to protect its data on a computing device from malicious local software, and prove its integrity to remote entities.

Trusted computing has many potential applications in the area of pervasive computing, a field where environments are rich in networked computing devices integrated with the environment. However, current trusted computing designs suffer from their dependence on centralized public key infrastructure and software monocultures. A public key infrastructure is required that accommodates the unique challenges that pervasive computing brings to trusted computing.

A public key infrastructure for trusted computing is described, based on the Pretty Good Privacy model of multiple signatures, and extended to be suitable for trusted computing. The concept of configurations is added, as well as a mechanism for sharing verifications of signature chains between trusted devices running this protocol. Two pervasive computing scenarios are described where such a system would prove valuable: a remote user making use of a SmartSpace, and an ad-hoc networking environment.

It is concluded that such an infrastructure could prove invaluable in a pervasive environment, and that a detailed investigation and implementation would be valuable in determining the effectiveness of such a system.


# 1 Introduction

*Pervasive computing* (PC) is a term describing a vision of numerous computing devices spread throughout the environment, so well integrated with the environment that the users of such devices barely even perceive that they are interacting with computers. Implementation of this vision combines elements of distributed and mobile computing with novel form factors and a device count orders of magnitude larger than currently encountered.

One requirement of pervasive computing will be that of security. Devices from a mix of administrative domains will wish to interact with each other and make use of the resources of other computing devices, while not having to sacrifice access controls, authentication, and

privacy.

A technology that could improve security in a pervasive computing environment is *trusted computing* (TC). This new research area explores the creation of hardware/software combinations that can verify to remote parties the state of the system, and protect data while on that system, while still retaining the flexibility of allowing arbitrary software and hardware to be used with each computing device.

One key feature of trusted computing is that of remote attestation: allowing a device to certify to a remote party that it is running a certain mix of software and hardware in a particular configuration. To provide this functionality, a hardware module capable of storing measurements regarding the operation of the device and its software is required. This device must be able to report these measurements, and sign them such that a remote party is assured that the measurements were in fact reported from a hardware measurement device. However, for the remote party to make use of these measurements, it must be able to verify that the signature on the measurements came from a valid hardware module, and also determine whether the configuration is suitable for a particular application.

The common solution to verifying hardware modules is the use of centralized public key infrastructure. Trust is universally given to a selection of central authorities to certify hardware measurement modules, and can use their signature to validate the public key of a hardware measurement module. The other challenge, that of determining the suitability of a set of measurements, is largely unaddressed, and normally assumed to be left as a challenge for the application developer.

Centralized PKI approaches are generally inappropriate for pervasive computing environments. In many mobile wireless environments, continuous connectivity with central infrastructure cannot be assured. It is also unlikely that all users of such a system would be able to agree upon a uniform set of authorities in which to invest trust to validate hardware. The diversity of software and hardware in a pervasive environment is such that the problem of determining valid configurations must be addressed in the infrastructure.

This project proposes a distributed public key infrastructure designed for verifying trusted computing remote attestations in a pervasive computing environment. The system addresses the issues of verifying a remote attestation to have arrived from a valid hardware module and determining the suitability of a particular configuration for tasks. Section 2 will discuss related work. Section 3 will briefly describe the area of trusted computing, and the particular

challenges of trusted computing as applied to pervasive computing. Section 4 will describe the proposed PKI architecture. Section 5 will describe two example scenarios of the architecture in action. Section 6 will conclude and suggest future direction for research.

## 2 Related Work

Much of the foundation work on trusted computing comes from the Trusted Computing Group[1] (TCG), a group formed from the remnants of the defunct Trusted Computing Platform Alliance (TCPA). The TCG, adopting and revising the specification off the TCPA, have proposed an architecture[2] for trusted computing systems, including a trusted hardware module (TPM), and low level software APIs required to interact with such a system.

While remaining intentionally vague on the public key infrastructure requirements of the system for remote attestion, their architecture does assume the existence of a 'privacy certificate authority', responsible for generating an identity certificate for a public key generated by a TPM. The privacy CA is assumed to be able to associate a public key sent with an identity request with a unique TPM, and if so, will sign a key for use by that TPM. The test of a valid TPM devolves to determining trust in the privacy CA that signed the key the TPM uses.

Work by members of the TCG have proposed[3] an extension to the use of a privacy CA through the use of direct anonymous attestation. This cryptographic technique allows a TPM to provide an attestation signature without having to provide the actual key used, while still allowing the receiver to determine the signature is valid. This eliminates the threat of a privacy CA colluding with verifiers to uniquely identify TPMs. It also makes it practical for the verifier to provide its own privacy CA.

Pretty Good Privacy[4] (PGP) was a tool created by Phil Zimmermann for the protection of emails. It sparked the concept of the 'web of trust'[5], where trust in a key is established by a chain of signatures between a known key and an unknown key. The primary difference in the web of trust model is that public keys can be signed by multiple signers, as opposed to the hierarchical model popularized by X.509 where every key is 'issued' by exactly one certificate authority. This change also allowed the flexibility for users to individually choose signers to trust, while still allowing for verification of most keys. PGP implementations of

the web of trust have typically not allowed for chains of trust greater than one signature long.

Many aspects of this model are borrowed for the presented PKI, namely the use of multiple signatures on keys and the local choice of trusted signers. However, unlike PGP, a signature in this case represents an authorization rather than verifying the authenticity of the key itself.

Unfortunately, very little thought has been put into the PKI behind TC in general. Previous efforts, particularly from Microsoft's Palladium initiative, raised significant distrust in TC as a whole, and in particular in any specification assigning trust anywhere but the end user. This has led many groups to avoid the question entirely, or simply refer to the corporate wired office, where local PKI may already be available and sufficient for the task: "the corporate environment will actually allow a very easy "local" implementation of remote attestation, without the need for a large scale PKI."[6]. Unfortunately, the corporate computing environment is largely at odds with the requirements of pervasive computing.

One proposal[7] reverses the question, suggesting the use of trusted computing to distribute the tasks of a hierarchical certificate authority. In this system, a certificate authority can delegate certificate issuing to an applet on a trusted computing host, reducing the bottleneck on central infrastructure. This is a useful technique for solving general PKI issues, but does not address the PKI needed for the authentication of the trusted computing environment itself.

# 3 Trusted Computing

Trusted Computing combines efforts from industry and academia to provide security assurances similar to those of closed computing platform, where only authorized software can execute, while still maintaining the openness and flexibility of an open computing platform, where nearly infinite mixes of hardware and software are supported.

Four key features are required for trusted computing:

- Process Isolation: The system must protect a trusted program from interference or memory inspection by all other programs on the system, including the vast majority

of the operating system.

- Secure I/O: Trusted programs must be able to read input from the user and present output to the user without fear of other programs (and even some devices) interfering with or monitoring the process. This normally means end-to-end encryption with input/output devices, and protecting memory buffers.
- Secure Storage: A program can store data to an insecure device like a hard disk in such a way that no other program can read or make alterations to the data. If the software configuration has changed since the data was last accessed, further access to the data must be denied.
- Remote Attestation: The platform prove to a remote entity that it is running a trusted computing environment, with the software in a particular configuration.

One of the key industry initiatives in trusted computing is that of the Trusted Computing Group (TCG). This group, a derivative of the disbanded Trusted Computing Platform Alliance, consists of a half dozen of the largest information technology providers, and dozens of other corporate contributors. The goal of this group is to issue standards for trusted computing hardware, specify software interfaces for interacting with this hardware, standards for evaluating both of these, and implementation details for a variety of platforms.

A key specification issued by the TCG is that of the Trusted Platform Module (TPM) a cryptographic processor designed to satisfy the need for secure storage and remote attestation. The device serves as a capable cryptographic processor, capable of performing at least RSA public key encryption/signing, AES symmetric encryption, SHA-1 hashing, and random number generation. Non-volatile memory stores the Storage Root Key (SRK), a protected symmetric encryption key used to protect larger sets of storage keys maintained in software, and the basis of secure storage. The endorsement key (EK) is asymmetric keypair, with the private key protected within the TPM, and used to sign requests for identity keys. The TPM can store several identity keys, which are asymmetric keypairs with the private keys protected inside the TPM. These identity keys are used for remote attestation.

Combined with an appropriate BIOS, the TPM also serves as a root for recording measurements. The TPM has a set of at least sixteen platform configuration registers (PCR) in which software may record measurements. Each measurement 'event' is added to a log. The event is prepended with the old PCR value, which is then hashed and stored back in

the PCR. Secure storage and attestations will both base their operations on a selection of platform configuration registers. Secure storage encrypted under one set of PCR values will not decrypt if the PCRs have changed (meaning the software configuration has not changed). Trusted software performing remote attestation will choose a set of PCRs to report to the remote entity, which will compare them to a set of trusted configurations.

**The proposed PKI assumes a hardware module in place similar to that of TCG's TPM, but should be implementable with any hardware module which provides equivalent features. We will refer to any such module as a TPM for convenience, though it is in no way meant to imply dependence on the specific standard of the TCG.**

## 3.1 PKI, TC, and PC

A key aspect of trusted computing is that of remote attestation. With remote attestation, the trusted hardware reports measurements of the current hardware/software state, signs this with a key protected within the hardware, and then presents this to software, presumably for it to send to a remote party. The remote party can verify the signature on the state, and then make its own determination as to whether or not the state reported is one it is willing to trust.

There are two key challenges here for the remote party attempting to verify the attestation:

- How does the party receiving an attestation verify that the signature on the state is in fact a signature from a valid TPM, and not just an arbitrary piece of software?
- How does the party receiving an attestation decide on whether the reported state is considered acceptable for an application?

The first challenge normally is assumed to be solved through the use of hierarchical PKI. The public key paired with the private key used to sign the attestation is issued by a trusted third party, who have their own methods of verifying the TPM (probably a whitelist or blacklist of TPM endorsement keys). The party receiving an attestation can then simply check the issuer's signature, reducing the problem to determining the trustworthiness of the issuer.

The second challenge is largely unsolved. It is assumed that the remote party will code this decision into application level logic, likely only trusting small combinations of software, or only relying on a limited set of measurements.

In a corporate environment, these problems are easily solved. Corporate PKI can verify and sign TPM identity keys, acting as the trusted third party. Valid configurations can also be made available centrally, based on what the corporation approves. This list can be updated as the corporation approves software patches or hardware upgrades; the number of approved configurations will be relatively small if the corporation forces standardized technology deployments. The participating devices are mostly wired PCs or equivalent laptops on high-speed wireless links, where power and bandwidth are not particularly scarce.

Many of the assumptions of a corporate environment break down in a pervasive computing environment:

- There is no one authority that all groups will trust. Entities will trust different parties, who may or may not be able to establish trust between each other. Not all devices on any individual local network will necessarily be under the same administrative domain.
- The space of possible configurations will be very large. It will be impossible for every device verifying attestations to maintain a list of all valid configurations while also allowing a useful subset of devices use the provided service.
- Global connectivity to centralized PKI servers may be sporadic or entirely unavailable.
- Some devices will be heavily constrained in power, bandwidth, storage, and processing abilities, while others may have huge excesses of these resources.

It is clear that a pervasive computing environment has unique challenges that need to be addressed before remote attestation can function.

# 4 Architecture

## 4.1 Elements

### 4.1.1 Public Keys

Private keys are (and must be for many remote attestation applications) still held in the TPM, and public keys paired with these may also be placed there. However, all the public keys, including the ones stored in the TPM, and public keys used for trust calculations are stored on a keyring maintained in software (see 'Keyring' below). The keyring holds the public keys matched with any private key in hardware, the public keys of known trusted parties, and any public keys obtained by communicating with other devices.

Keys have minimally the following elements.

- A fingerprint: an ID, or unique identifier of some sort. This does not have to equate to any form of identity, rather is simply a concise form of referring to the key.
- A public key of a key pair.
- A set of signatures (see 'Signatures' below).
- A flag representing the source of the key: from a known private key pair, downloaded explicitly by the user, or obtained automatically from a peer.

The key may optionally have further information such as an entity name or contact address, which will be useful for human trust of the system (eg. a human may trust "International Business Machines", but not necessarily the same key shown only as "key ID 0x1234ABCD"). This is purely optional, since the PKI does not require human authorization of trust by name except for the initial set of trusted roots.

### 4.1.2 Configurations

Trusting that the trusted hardware operates correctly is not enough to trust a system for a specific purpose - the configuration of hardware and software reported by the trusted hardware is equally as important.

The format used for a configuration by the TPM specification uses a set of platform configuration registers (PCR) to represent the state of the system, backed by a log. Each register has a specific intended purpose (eg. one is reserved for BIOS measurements). When a measurement is made, this measurement is written to the log. The measurement is prepended with the current PCR, hashed, and stored back in the PCR. This means a PCR and a log verify each other, since the log can be rehashed to verify against the PCR value.

A configuration should specify minimal requirements to make a trusted module appropriate for a specific task. The simplest form is to specify sets of PCR:VALUE pairs that are valid, while a more complex form would analyze a sequence of measurements.

Each signature on a configuration is an assertion by the signer that the signature is valid for specific tasks, specified by text keywords. These keywords should be agreed-upon between the trust provider and the clients who choose to incorporate it into their applications. Different signatories may sign the same configuration with different keywords; the signatures do not interact with each other.

### 4.1.3 Signatures

A key difference between this PKI and a X.509 hierarchical PKI is the difference between a key signature and a certificate issuer. In a X.509-style PKI, a certificate authority issues a public key certificate, combining the public key with a signature from the certificate authority. Each certificate has exactly one issuer. The PGP style of PKI has a public key issued by the holder of the matching key: a'self-signature'. Anyone who verifies the authenticity of this key through out-of-band means may add their signature to it. Regardless of PKI style, a key is verified as authentic by verifying a path between a trusted key and the key in question through a sequence of signatures.

In the X.509 model, trust is given to a very specific set of public keys, whose public component is stored in each device. The PGP model uses the concept of a local signature to verify a key's authenticity: a signature made with a key whose private counterpart is known is given absolute trust. The PGP model also allows the user to specify on a per-key basis how much trust is given on signatures that key itself makes, allowing each user to choose who to allow to assign trust transitively.

Our PKI model resembles that of PGP, where keys (and configurations, though this concept

is not present in PGP) can be signed multiple times. Also like PGP, trust is rooted in locally-signed keys.

Unlike both the PGP and X.509 models, a signature asserts what actions a key is authorized to perform, rather than its authenticity. With the exception of human-readable information, there is no authentication required of a key, since there is no machine-interpreted binding between a key and a particular identity. The signature instead specifies the following attributes about a key.

- A flag, set if the key is a valid public key whose private key is protected in a TPM. Present only on key signatures.
- A set of keywords representing what services the signature authorizes. On a key, this authorizes the key to sign configurations for this keyword. On a configuration, it authorizes the particular configuration for use with this keyword. One special keyword represents configurations authorized to implement this PKI, and relates to transitive trust.
- A flag, set if the key is trusted to sign other keys as valid TPMs.
- A flag, set if the key is trusted to sign other keys as valid signers of TPMs.
- A flag, set if the key is trusted to sign keys to authorize them to sign configurations with the same or less keywords as granted this key.
- A flag, set if the key may grant the key configuration signing ability to other keys.
- A set of key fingerprints describing a signature chain. This attribute specifies that the signer itself has not authorized this action, but has determined that such action is authorized through the trust chain specified by the sequence of keys. If not specified, then it is the signer itself that has authorized the actions.

One special type of signature, called a *local signature*, is a signature made by a private key contained in the local TPM. Local signatures serve as the root of trust for the PKI: the authority that a signature assigns is ignored unless the key making that signature has been given the permission to assign such rights from a local signature directly or transitively.

**Revocation:** There are several cases where a signature may need to be revoked. For example, a configuration that was initially trusted to be secure may later be found to be insecure, or a key that was trusted may later be extracted from hardware and need to be revoked. Signatures are revoked by creating a *revocation signature* that cancels out the matching signature. A revoked signature does not devalue any of the other signatures on the key. To revoke a key in its entirety and nullify all of its signatures, a special revocation signature must be made by the private key matching the key being signed. Configurations themselves can never be revoked, since a configuration is not a security attribute and therefore cannot possibly be 'insecure'.

### 4.1.4 The Keyring

The keyring is name given to a store of public keys, configurations, signatures, and revocations. There may be many keyrings on a device, corresponding to the different trust decisions made by different users, though in a pervasive computing environment there will likely only be one keyring active on a device at any instant. If there are multiple keyrings, each must minimally contain the public counterparts of every private key in the TPM.

A trusted module maintains the keyring to ensure that malicious processes do not corrupt it. This also allows some key optimizations. If the module managing the keyring refuses to save an invalid signature, the signature itself needs only be tested when first received. This becomes even more useful for local signatures, which will not be verified at all if generated locally; this will be a very common operation, as all trust is derived from local signatures.

The keyring has a fixed upper size limit appropriate for the device. This must be at least large enough to store the public counterparts for every private key in hardware, as well as any public keys from trusted authorities pre-imported into the device. Other keys, as well as configurations, signatures, and revocations are obtained opportunistically by communicating with other devices until the keyring is full. If more space is required, the keyring is responsible for discarding elements it doesn't need using an appropriate replacement policy. Keys with private counterparts, and keys which were explicitly downloaded are never discarded, and local signatures on these are never discarded except where the object it signs is discarded. Revocations are discarded as late as possible, and never before the matching signature is discarded.

## 4.2 Sharing the Keyring

The full content of the keychain can be made public to any software or remote device interested in viewing it. This is used to spread key and configuration knowledge relevant to the local environment to devices.

A remote device can send (either targeted or broadcasted) a request for information. This is protocol-dependent, but several possible requests could be envisioned.

- Request for a specific key, identified by fingerprint.
- Request of a specific node for the public counterpart of the key it intends to use.
- Request for a configuration matching a specified set of PCRs.
- Request for signatures on a key or configuration, optionally with a list of fingerprints of trusted authorities that the signature should eventually lead to.
- Request for keys and/or configurations relevant to the current environment.

The ability to volunteer information unsolicited is also important. One key use of this will be revocations: a device should offer revocation signatures whenever it sees another device reference a key that has a revocation on it. This ensures revocations are heavily propagated. Keys, signatures, and configurations can also be spontaneously offered. A keyring is never required to store material it receives, for example if it is full and believes its other data more relevant, if believes the data to be incorrect, or simply believes the data is not worth the computational effort to verify.

Privacy may be of some concern in cases where the choice of trusted keys and configurations is deemed sensitive information. This can be handled through implementation, with the PGP 'non-exportable' signatures as a useful example. In general, limitations on key sharing reduces the effectiveness of the system, so should be avoided except where absolutely necessary

### 4.2.1 Trusted Signature Chains

The list of fingerprints in signatures is used to indicate that a key has been approved for rights based on the authority of another trusted key rather than the signer itself. It represents an assertion by the signer that it has verified the signatures of the chain described by the list of fingerprints.

A malicious node could easily fake such a list, so such signatures are in general not useful and should not be accepted into the keyring. However, if the node providing such a signature is in fact a trustworthy device in a configuration appropriate for verifying chains, it is then known for a fact that the chain was verified. In which case, the signature can be imported. If the receiver of the signature is also trusted, it can share this signature in the same trustworthy manner, since it is known that the trusted device only accepted the signature chain from another trusted device.

## 4.3 Reducing Trust Chains

One key feature of this PKI system is the ability to reduce trust. It is inefficient in time to verify an entire signature chain for every trusted operation, and inefficient in space to store all the relevant keys and signatures. Once a signature chain is verified back to a locally-signed key, the local device can record the list of fingerprints back to this key, and then sign it with a key of its own, attaching the list of fingerprints and the final set of permissions the original signature grants the key.

For the local node, the local signature is sufficient to authorize the key for actions, and the rest of the chain never again needs to be verified. The remainder of the intermediate keys and signatures can potentially be discarded if space is needed in the keyring.

This local signature contains the list of fingerprints of the original signature chain for the case where the signature is later shared. If the node is trusted, remote nodes can be sure that the chain is accurate, meaning they do not have to verify the chain themselves, nor even obtain the intermediate keys.

# 5 Example Scenarios

The following two examples describe pervasive computing scenarios where this PKI could potentially increase the effectiveness of trusted computing.

## 5.1 The SmartSpace

A SmartSpace describes the idea of an environment heavy with computing infrastructure with which users interact. The idea was described (not by name) in an article[8] by Mark Weiser, outlining both imagined scenarios, and early implementations attempted at PARC. A classic SmartSpace scenario is the office conference room, where everything from the data projector to the lights are networked and can be controlled from the personal computing devices of those inside the conference room.

A use for trusted computing in such a scenario might be an employee of company A giving a confidential presentation to an executive of company B under non-disclosure agreement. Employee A wishes to display data on the projector from his PDA, but prevent unscrupulous network administrators in company B from making a copy of the data while it is displayed there.

The PDA wishes to ensure that the projector is running trusted hardware, and is in a configuration that will protect the data while displayed. The PDA only trusts A's key server, and does not have the keys or signatures in memory to determine if B's projector is in fact trustworthy.

The PDA requests an attestation of the projector, which the projector provides. Not being able to verify the attestation, the PDA broadcasts a request for the appropriate key and configuration, asking for signatures which can be chained back to company A.

A (very large and powerful) company B key server in the in infrastructure detects the request. It looks through its database, or even contacts the Internet if necessary, to build a signature chain between its own hardware/configuration and company A. It finds this and sends it to the PDA. The PDA verifies the signature chain, then locally signs the key. The PDA can then trust B's key server for verifying signature chains. The key server can then use its extensive keyring to construct a signature chain between company A and the projector (or

any other element of company B's infrastructure), and send a local signature with the chain listed only as fingerprints. The PDA does not need the full chain, and does not even have to verify the signature it is sent, because the key server is trusted.

Note, in this scenario, company A does not have to trust the key server of company B to sign keys. It needs only to trust through some signature chain the trusted hardware that the key server runs and the configuration of the key server software. B's key server can then be trusted to verify signature chains in a trustworthy manner.

As an additional bonus, the key server can also establish trust in the reverse direction, ie. from the infrastructure to the device. The key server requests the key the device will be using, and for any relevant signatures its carrying. The key server then verifies the trustworthiness of the trusted hardware and configuration, then signs the key (on its own authority, not with the fingerprint IDs), and offers that signature to the device. The device can then use this when communicating with other devices in the infrastructure, which presumably already trust B's key server.

## 5.2 The Ad-hoc Network

An ad-hoc network is one formed from wireless devices in the local area. Practically the opposite of the SmartSpace, an ad-hoc network has limited or no infrastructure, and the devices may be from a wide variety of administrative domains. Most of the devices will be wireless, and be limited in storage space and computational power.

In such an environment, one may desire trusted computing, even to establish membership in the network. An untrusted node may 'cheat', sending its own data through the network, while dropping packets it is asked to forward to conserve its own bandwidth. A trusted node can be verified to forward packets fairly.

It is unlikely that any one node will have a large enough keyring to establish trust with all of its neighbours. However, it can broadcast requests for keys to establish this trust (assuming such requests are always allowed, regardless of trust). Other nodes, if they have useful signatures and keys, can provide these. Even better, once any pair of nodes establish mutual trust in each other, only one of the pair has to evaluate a signature chain, and can then share its results with others.

The local signing of keys creates a stronger local root of trust, where trust can be verified in many cases with a single signature. Where this is not possible, a node can make use of the combined keyrings of every other node if needs be to establish trust. This becomes even more effective where nodes are mobile and may occasionally obtain Internet access. This could allow the construction of what would essentially be a delay-tolerant PKI, though speculation as to how such a system would operate is left for future work.

# 6 Conclusion

The proposed PKI infrastructure addresses many challenges for trusted computing in the pervasive environment. By sharing keys, the need for continuous Internet connectivity is greatly reduced. The concept of a fixed-size keyring allows small devices to bound the amount of storage that is required. By relying on the trust that can be generated in a remote attestation, the number of cryptographic operations that need to be performed by some devices are greatly reduced, and can be organized such that the majority of the operations are performed by the most capable host.

Adding configurations to the PKI itself has helped address the problem of the wide diversity of devices in pervasive computing environments, and allows for the same infrastructure to verify all aspects of trust in a remote attestation.

Overall, this PKI can solve many of the challenges that trusted computing faces.

## 6.1 Future Direction

Many assumptions are made regarding both the composition of a pervasive computing environment and how trusted computing would be used in such an environment. Various real-life pervasive computing environments should be studied, to determine the security requirements and determine whether or not all the challenges of pervasive computing described are actually problems in practice. Without this, it will be difficult to evaluate the actual usefulness of this approach.

Mathematical models of the system would be useful in determining the storage requirements in the keyring. It would be very relevant to know how large a keyring is required in var-

ious scenarios to efficiently maintain trust in a network, and how much communication is required for each size. Also, the actual replacement policy of the keyring should be studied to determine an ideal method of choosing which objects to delete.

An implementation would answer the most questions about the system, forcing decisions on details, and revealing potential problems in both the protocol and the scalability of the system.

# References


[1] "Trusted Computing Group: Home," 2004; http://www.trustedcomputinggroup.org/

[2] *TCG Specification Architecture Overview,* rev. 1.2, Trusted Computing Group, Apr. 28, 2004.

[3] E. Brickell, J. Camenisch, and L. Chen, "Direct Anonymous Attestation," tech. report HPL-2004-93, Trusted Sytems Laboratory, HP Laboratories Bristol, Jun. 3, 2004.

[4] "Welcome to The OpenPGP Alliance," *OpenPGP Alliance*; http://www.openpgp.org/

[5] A. Abdul-Rahman, "The PGP Trust Model," *EDI-Forum: The Journal of Electronic Commerce,* Apr. 1997

[6] "Interested Uses of Trusted Computing, Part 2," *Unlimited Freedom*, Mar. 22, 2004; http://invisiblog.com/1c801df4aee49232/

[7] A. Dent, "Distributing PKI functionality using trusted computing," presentation at *Open Workshop on Trusted Computing,* Mar. 2004.

[8] M. Weiser, "The Computer for the 21st Century," *Scientific American*, vol. 265, no. 3, pp.94-104, Sept. 1991.