Deploying the Trusted Platform Module (TPM) to Increased Fairness and Trust in P2P File Sharing Servents

JOERG ABENDROTH* and JEAN-MARC SEIGNEUR#
*Corporate Technology, Siemens AG, 81739 Munich, GERMANY
# University of Geneva, SWITZERLAND

Abstract:
A number of P2P file sharing systems have tried to rate the level of contribution and participation of peers in order to avoid free-riding by selfish peers and increased performance and reliability of file sharing among all peers. However, the proposed software rating mechanisms (including the ones that use computational models of trust) are still flawed by the use of modified peer servent applications that bypass or trick the rating mechanism. In this paper, we propose to bind the peer servent application with the new hardware Trusted Platform Modules (TPM) that are going to be massively deployed. In doing so, it is not possible anymore to bypass or trick the computational trust and rating mechanisms associated with the peer servent application version and P2P file sharing can trustworthily use these mechanisms for increased reliability, fairness and performance. To validate our approach, we detail how the current specification of TPMs can be used in P2P file sharing to enforce trustworthy rating mechanisms.

Key-Words: - Behavioral Identity, TPM, TCG, Computational Trust, P2P, PCR, AIK

1 Introduction

A number of P2P file sharing systems have tried to rate the level of contribution and participation of peers in order to avoid free-riding by selfish peers and increased performance and reliability of file sharing among all peers. For example, the official KaZaA [18] peer SERver/servent (servent) application version computes a user participation level. A user participation level increases each time the user rates the integrity of a file, increases the number of megabytes shared with the peers community, adjust bandwidth, and so on. Some servents have better behavior than other and users are free to use any servents. The P2P network community has the option of treating selfish servents different than well behaving.
However, the proposed software rating mechanisms (including the ones that use computational models of trust [25,26,27])) are still flawed by the use of modified peer servent applications that bypass or trick the rating mechanism. It is well-known that modified KaZaA servent's application versions exist and trick the user participation level rating mechanisms.
In this paper, we propose to bind the peer servent application with the Trusted Platform Modules (TPM) that are going to be massively deployed. In doing so, it is not possible anymore to bypass or trick the computational trust and rating mechanisms associated with the peer servent application version and P2P file sharing (or other recommendation based systems) can trustworthily use these mechanisms for increased reliability, fairness and performance. Our system reliably allows the system to differentiate different servent versions and to enforce correct reporting of servent behavior. The servents of the P2P file sharing system can rely on industrial standardized TPM hardware to identify the servent versions and to differentiate between fair and unfair behaving servents versions. To validate our approach, we details how the current specification of TPMs can be used in P2P file sharing to enforce trustworthy rating mechanisms.
The following Section 2 describes computational trust systems. Section 3 presents the design of a TPM enhanced P2P file sharing system. Then Section 4 describes a lifecycle and gives an example, afterwards Section 5 evaluates the system. After a discussion of related work in Section 6, we draw conclusion in Section 7.

2 Computational Trust Systems

Computational trust systems have been researched for some time [28, 7]. Figure 1 gives an overview of a trust engine. The decision-making component can be called whenever a trusting decision has to be made. The Entity Recognition (ER) module [29] bridges the gap between identity management and reputation by recognizing the entities involved in the interactions with attack resilience and privacy protection considerations. The decision-making of the trust engine uses the trust module to dynamically assess the trustworthiness of the requesting entity and evaluates the risk involved in the interaction based on the available trust and risk evidence in the evidence store. A computed trust value in an entity may be seen as the digital representation of the trustworthiness or level of trust in the entity under consideration.

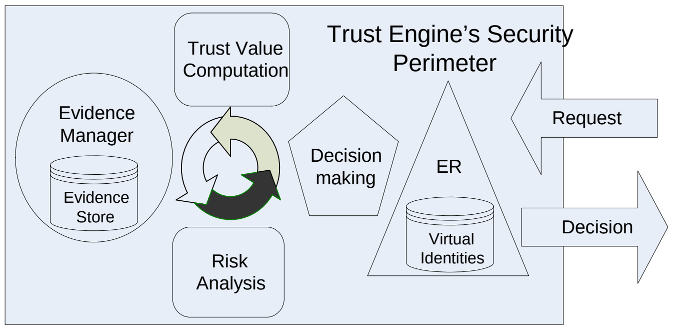

Figure 1. Overview of a Trust Engine

Successful real life systems include eBay, Amazon or the moleskiing website [6]. However these systems are centralized and it is hard to transfer their mechanism to decentralized systems. Often the main problem is that no reliable enforcement can be done on the decentralized servents, which is the case in P2P file sharing. For example the Sybil attack [30] consists of the use of multiple faked identities used by the same user to vouch for several successful faked interactions.

A trust metric [29,31,12] consists of the different computations and communications which are carried out by the trustor (and his/her network) to compute a trust value in the trustee. Different trust metrics have been researched to achieve greater attack-resistance. However, none so far are perfect in fully decentralized systems, such as in P2P file sharing.

3 Design of a Strong Computational Trust System

The system proposed in this paper mitigates the attacks described above by means of using a TPM as the underlying decentralized, trusted third party. The TPM vouches for the behavioral identity. Our framework prevents attackers to act nicely today and "turning bad" tomorrow.

3.1 Utilizing TPM Hardware to Build Strong Trust

The Trusted Computing Group (TCG) standardizes the trusted platform module (TPM)[3,4,5], which is already used today in commercial applications to provide a root of trust, secure key and data store [24, 1, 2]. An extensive deployment of TPM-aware computers can be foreseen because modern operating systems require or use the TPM chip [14, 17, 15, 16]. Two benefits are provided by the TPM: (1) secure storage of key material, that may only be accessed in certain platform states or by certain persons; (2) provide a trustworthy source that can be queried by external entities. It is important to note that these benefits can only be obtained if the device software (and OS) is TPM-aware. Care has to be taken to design a system that inherits the trust placed in the TPM chip and that does not allow attackers to subvert the implementation.

A TPM is a passive chip that needs to be accessed and used by the operating system software. The software and user can choose to use or not use the TPM chip (e.g. it can be deactivated). If it is used, the chip has to be involved in the boot process from the first step on. The root of trust has to ensure that the TPM is fed with the hash values of the first program to run, which is the root of trust itself. This requirement is ensured by means out of scope of this paper, but already existing in the market place [13]. The second requirement is that the root of trust has to hash and store evidence of the second program in the boot process in the TPM chip, then control can be handed over to that program. At the point the second program gets control of the computer there are three abstract things that can happen: (1) First it can discontinue using the TPM (e.g. switch of the chip); (2) it can do things that it is not expected to do, namely hand over control without first storing evidence of the new controller in the TPM; and (3) the program acts as expected continuing the boot process involving the TPM and correctly sending evidence to the TPM. We will review the consequences of the three cases below.

In the case where the program turns the TPM off, the chip can not be used to provide trustworthy evidence of the existing programs, neither can it be switched on again later. Hence the case where the TPM is turned off is of less interest to us.

In the second case where the second program in the boot process loads and hands over control to another program without first storing evidence in the TPM is similarly undesirable. In this case it would not be possible from the outside to differentiate between the different programs that could be started by the second program in the boot chain, hence it can not be differentiated if a well-behaving or misbehaving program was started - the system security can be said to be broken. Although the TPM is not switched off, it will only "know" the existence of the programs up to the second program in the boot chain. Any actions done by the following, not recorded, programs will be counted as the actions of the second program. In practice, the second program, which did not honor the TPM boot process, would have already been counted as not trustworthy. The systems with a TPM reporting that second program is running should be counted as not trustworthy[1].

[1] Two definitions of trustworthiness are used in this paper:
I)Trustworthiness based on the TPM hardware (binary either trustworthy or not trustworthy)
II) Trust values that are computed based on evidence and a trust metric (e.g., a value between 0 and 1)

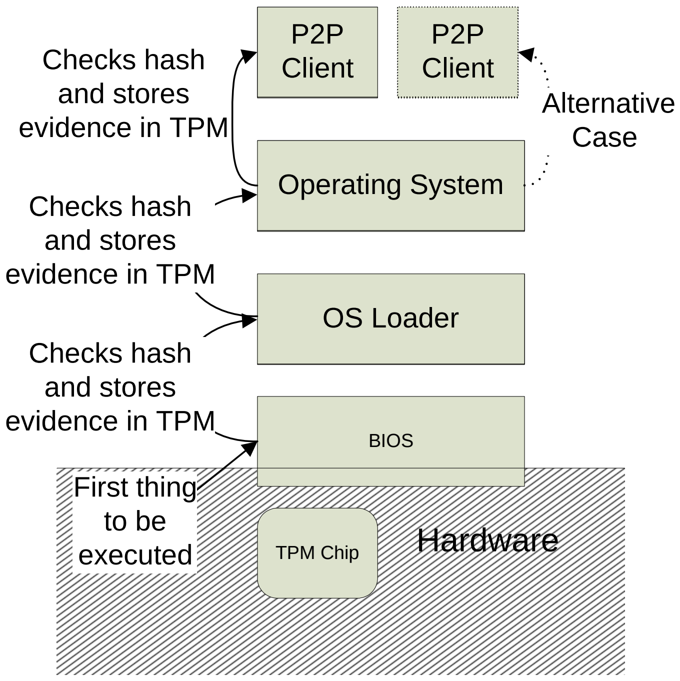
P2P Client
P2P Client
Alternative Case
Checks hash and stores evidence in TPM
Operating System
Checks hash and stores evidence in TPM
OS Loader
Checks hash and stores evidence in TPM
BIOS
First thing to be executed
TPM Chip
Hardware

Figure 2: Boot Sequence

The third case is the one depicted in Figure 2. In this case evidence of all programs up to the last (currently executing) program is recorded in the TPM. It is possible to request a signed statement from the TPM about the currently loaded (and running) programs. The figure depicts the root of trust, the OS loader, the Operating System, and one P2P servent. Here evidence of each state of the boot sequence is stored in the TPM (i.e. platform configuration registers, PCRs) and hence it is possible for an outsider entity to identify the behavioral context.

3.2 Behavioral Identity

We define a behavioral identity by the executable code in the boot sequence. It is expected that each behavioral identity can share the same trust value "account", meaning all actions and recommendations will be accumulated to the same trust value.

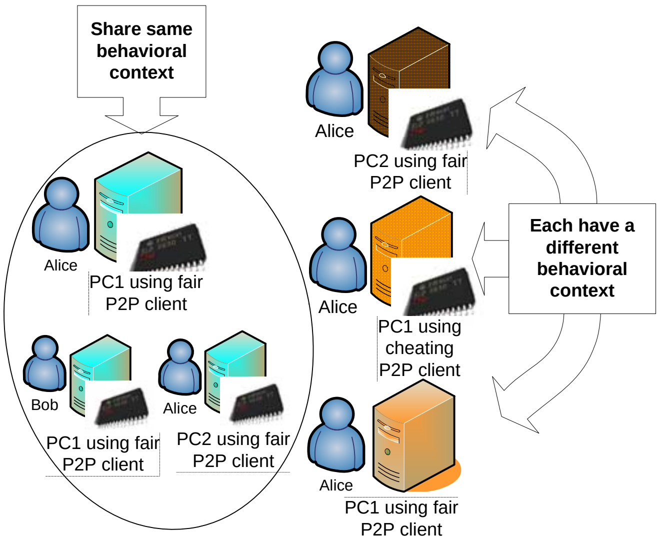
Share same behavioral context
Alice
PC1 using fair P2P client
Bob
PC1 using fair P2P client
Alice
PC2 using fair P2P client
Alice
PC2 using fair P2P client
Alice
PC1 using cheating P2P client
Each have a different behavioral context
Alice
PC1 using fair P2P client

Figure 3: Behavioral Identities

Figure 3 shows that the same PC with the same software configuration will be counted as the same behavioral identity and thus accumulate trust in the same trust account. Hereby it will not be necessary to differentiate between the different users of the same configuration. Our assumption is that the user can not influence the behavior of the programs (the next section explains this assumption in more detail). However several parameters can cause a change in behavioral identity: a different PC configuration is used (e.g. OS version) is used, or programs any of the programs started with the servent changes. It is important to note that although the user can freely choose the configuration (P2P servent), that he can not cause an attack by choosing one identity to acquire trust and later to choose another identity to take advantage of the accumulated trust. Furthermore, as the configuration is known to others (e.g. the TPM may report sufficient information so that it can be checked which servent version is used) it can be noted the system is PC/User independent. Hence it is possible to have the servent configuration building up trust independently of the user or the PC.

3.3 Behavioral Variance

The system of behavioral identities works well for fixed programs, which do not allow the user to influence the program behavior, e.g. by setting unfair configuration parameters. To cope with latter behavioral variance was introduced. Behavioral Variance expresses the flexibility allowed to a peer servent to change the behavior. Valid values are from 0 to 1, with 0 expressing a static behavior (no user defined parameters) and 1 expressing a very variant (or undefined) behavior. There are two ways of deriving this parameter: manually and automatic. Manually, means manually examining the program and defining a value, this is the preferred method. Automatic means taking the current trust value, determining if the observed behavior is outside the current variance and adjusting the value if necessary. There is room for research of the best algorithm for automatically finding the behavioral variance. In practice it is expected that the behavioral variance is also an indicator for how likely it is that the user can collect trust and later exploit the system.

3.4 Actions

In the following we propose a simple mechanism of actions, but the approach of utilizing TPMs to build enforced trust can be extended to other trust computation mechanisms too.

We define a set of actions, as shown in the table below:

A1	Successful file download
A2	Failure during file download (small loss)
A3	Virus after downloaded file execution (high loss)
A4	No further sharing of the downloaded files

Table 1: List of Actions

3.5 Identity of the User

Many computational trust systems include identities of the user [7, 6, 12]. In the basic version of our system, to rely on the user identities is not mandatory. However our framework includes identities bound to each TPM. The main purpose of this identity is to verify the authenticity of a TPM. The EK (Endorsement Key) is unique to each TPM and has special constraints in its usage (due to privacy concerns). The EK may not be used for signing or other operations than to retrieve attestation identity keys (AIKs) from trusted third parties [11]. The AIKs are used to verify the correctness of the behavioral identity (or PCR values as explained later).

The use of AIKs and behavioral identities rather than user identities has the advantage that more observations can be collected than if each user has its own trust account.

Although a user may still try to carry out a Sybil attack on the AIKs, it can be discovered, as the underlying EK is unique. As a countermeasure trusted third parties (that issued the AIKs) provide a function that can say whether or not several given AIK signatures belong to different TPM EKs. Hence if one interaction has been certified by the local TPM, the evidence used to build the trust value can not be used twice in the system as repetition will be detected.

In the case where user identities are needed it is possible to include them in the protocol, but the creation, distribution and protection are beyond the scope of this paper.

4 Lifecycle/Operation of the System

In this section we give an example of how the system operates.

4.1 Initialization Phase

The TPM of each servent generates an AIK, which is then certified by a trusted third party using the standard TPM AIK issuing protocol (e.g., the one described in [11]).

A trusted entity reviews the P2P servents (i.e., the configuration options) and determines the behavioral variance; this might also be done by the users themselves. The programs without reviews get a variance of 1 (full scope of behaviour variance).

4.2 Servent Booting

Alice boots her first PC, PC 1. The trusted BIOS is the first thing to be started and records its own hash in the PCR 1; then it records the hash of the OS Loader in PCR 1[2]. Alice continues to boot the OS and the fair P2P servent. The final value in PCR 1 is:

PCR 1 Alice: *000000003af29c6441*

Bob also boots his PC2, but uses the unfair P2P servent:

PCR 1 Bob: 000000007fcc31542c

Further on the following participants exist:

PCR 1 Charlie: 000000002144fcad3
PCR 1 Dorothy: *000000003af29c6441*
PCR 1 Ernest: 0000000021d42c6f1a

Dorothy also uses the fair P2P servent, in the same configuration as Alice. Charlie, Ernest, Frank and George use the unfair P2P servent. Ernest and Frank have the same configuration setup, thus both PCR values are the same.

4.3 Evidence Collection

Now our group starts to participate in a P2P network. We briefly list the interactions (see table 1 for meaning of the actions) and give the explanations afterwards:

Alice>Charlie: A1	Bob>Alice: A1
Alice>Dorothy: A1	Bob>Alice: A1
Alice>Frank: A4	Bob>Charlie:A4
Alice>Charlie: A1	Bob>Frank:A1
Alice>Charlie: A1	Bob>Ernest:A1
Dorothy>Frank: A1	Ernest>Charlie:A3
Dorothy>Alice: A1	Ernest>Bob:A2
Dorothy>Alice: A1	Frank>Charlie:A3
Dorothy>Frank: A4	Frank>Alice: A3
Dorothy>Bob: A1	Frank> Dorothy: A3

We observe that Alice and Dorothy have well behaving servents and many successful interactions, that are both counted into the same behavioral identity account. Bob has also a well behaving setup, but he is collecting trust only on his own. Ernest and Frank might know how to tweak the unfair servent to gain an advantage, but this causes both to collect negative feedback into their shared behavioral identity. The other participants are ignored in this example.

4.4 Recording and Transferring Evidence

For each interaction the servent records the action in a log file and submits it to the TPM, which records the action in a special PCR (one PCR per behavioral identity). Later the TPM can vouch for the PCR value and the log file allows one to reconstruct which actions were done.

The transfer of observations is done by the use of the TPM_PCR_Quote command, which provides the PCR values in a data blob signed by the AIK. The remote party can verify that the PCR values originate from a real TPM. Repeated transfer of the same observation is discovered as the signature will be from the same AIK.

Below an example token is given, that is transmitted if a remote party asks for the trust value for a certain behavioral identity.

(BI; [BV]; [UI]; (TV)signed by AIK; [Logfile])

[2] The resulting value is the result of a one-way function, thus it is not feasible to predict an extension value so that a desired PCR value is achieved.

BI: Behavioral Identity, the value of PCR1 after the servent is started.
BV: Behavioral Variance, the manually determined value how much behavior can differ
UI: User Identity, if required by the system
TV: Trust Value, the concatenation of the actions recorded in the PCR. This value is signed by the AIK of the TPM to vouch for the correctness.
Logfile: optionally a log file of the observed interactions used to compute TV.

4.5 Example of a Computational Trust System

After a few interactions George wants to interact with Alice. First he queries the P2P network. He retrieves the trust value for Alice's behavioral identity (*000000003af29c6441)* from each participant:

Alice says: +1 (the interaction with Dorothy)
Bob says: +1 (the interaction with Dorothy)
Charlie says: +4
Dorothy says: +2
Ernest says: 0
Frank says: 0 (not accepted transactions count 0)

George himself knows he has 0. He then computes the overall trust value and bases his decision on this value.

Then George wants to interact with Ernest:

Alice says: -1
Bob says: -2 (big loss counts double)
Charlie says: -2
Dorothy says: -1
Ernest says: 0
Frank says: 0

Ernest and Frank share the same behavioral identity and will be noted as not well behaving.

5 Evaluation

The presented system utilizes the TPM as an underlying trust provider. Unlike in other computational trust systems the trust of the TPM collected evidence is enforced by strong means. Remote parties can first verify that the behavior of the trust value reporting servent is trustworthy; then receive the trust value using the TPM_PCR_Quote command. The advantage of using the TPM command is that attacks relying on resubmission of the same evidence can not succeed, because each TPM has one or several unique AIK and the remote party can recognize if evidence of a particular AIK is already taken into account.
The use of behavioral identities has several advantages. First, behavioral identities are naturally recorded by the TPM usage proposed in current standards and practice and do not restrict the user.
Secondly, behavioral identities are user-independent and thus allow the collection of trust values to be quicker. It is even possible that new users benefit from already established knowledge that this servent is well behaving.
Finally, behavioral identities are naturally dividing different interaction domains and thus facilitate a clear division between interactions done in one setting (e.g. ebay trading) to another (e.g. P2P file sharing). Some computational trust systems inherit problems from the fact that a user may collect trust in one interaction domain (which is easier) and exploit it in another (where the benefit is bigger). In our approach behavioral identities are bound to the interaction domain that the respective servent program is limited to.

6 Related Works

The TPM specifications [3,4,5] have been subject to a wide range of research [19, 22, 23], some of it focusing on security aspects or privacy concerns.
Poritz and Cachin [20] describe the boot process, like used by our mechanism, but focus their research on logical errors in the TPM system design.
Closest to our mechanism are Sandhu and Zhang with their "Peer-to-Peer Access Control Architecture". They employ the trusted computing technology to build a decentralized access control architecture [21]. Unlike our system they employ a micro kernel approach [22] and a secure boot process, instead of allowing the user to start any kind of servent. We use the TPM to retrieve trustworthy evidence for the computation trust system, without restricting the user with regard to the applications that may be used.

7 Conclusion

We have presented a computational trust system that benefits from the trustworthiness of an underlying TPM system. Users can freely choose the servent version that they want to use to participate to the P2P file sharing system. However, the users can not lie about the servent version that they use. Over time trust is formed in the servent versions (i.e. behavioral identities) by means of recorded actions. Several users can share the same behavioral identity and thus trust in behavioral identities increases quite fast, interaction after interaction in the whole P2P network..
The proposed system is flexible enough to accommodate different computational trust algorithms.
Although we use the example of P2P file sharing, we also see applications of our approach in the area of recommendation-based contract negotiation, online auctioning or community knowledge sharing. All these systems can benefit from an enforced trust in the applications a user runs, in addition to solely build trust based on his past interactions.

References:

[1] Wave Systems Corp., Embassy Trust Suite Software Product , *Company web page,* (checked 11.8.06)http://www.wave.com/products/ets.html

[2] Wave Systems Corp., Trust System / Toolkit , *Company web page*, (checked 11.8.06) http://www.wave.com/products/toolkit.html http://www.wave.com/products/trust_system.html

[3] Trusted Computing Group, *TCG Main: Part 1 Design Principles*, Version 1.2 Rev.94, Standard Specification, 2005

[4] Trusted Computing Group, *TCG Main: Part 2 Structures of the TPMs*, Version 1.2 Rev.94, Standard Specification, 2005

[5]Trusted Computing Group, *TCG Main: Part3 Commands*, Version 1.2 Rev.94, Standard Specification, 2005

[6]Paolo Avesani, Paolo Massa, Roberto Tiella *Moleskiing: a Trust-aware Decentralized Recommender System*.. FOAF Workshop at DERI Galway, September 2004.

[7] T Grandison, *A survey of trust in Internet Applications*, IEEE Communication surveys Fourth quarter 2000

[8]Chuk-Yang Seng, William A. Arbaugh, *A Secure Trust Establishment Model*, pp. 78-85, IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing -Vol 1 (SUTC'06), 2006.

[9]S. Balfe, S.Li and J.Zhou,*Pervasive Trusted Computing*, Security and Privacy in Pervasive and Ubiquitous Computing, 2006

[10]W. A. Arbaugh, D. J. Farber, and J. M. Smith, *A Secure and Reliable Bootstrap Architecture*, IEEE Symposium on Security and Privacy , pp. 65-71, May 1997

[11]Verisign Inc., *Verisign Certification Practice Statement, Technical Report* Version 1.1, August 1996

[12]J. Golbeck and J. Hendler, *Accuracy of Metrics for Inferring Trust and Reputation in Semantic Web-based Social Networks*, presented at the 14th International Conference on Knowledge Engineering and Knowledge Management, 2004.

[13]Phoenix Corporation, *Phoenix Trusted Core Bios*, http://www.phoenix.com/en/Products/Core+System+ Software/TrustedCore/default.htm, checked 20.8.2006

[14]Microsoft Corporation, *Whitepaper: Trusted Platform Module Services in Windows Vista*, from website:*http://www.microsoft.com/whdc/system/platform/ pcdesign/TPM_secure.mspx*,checked 20.8.2006

[15] Open Trusted Computing Consortium, *website: http://www.opentc.net*, Part of the 6th European Commission Framework. Checked 20.8.2006

[16] European Multilaterally Secure Computing Base Consortium, *website: http://www.emscb.de/*, Partly funded by the German Federal Ministry of Economics and Technology, Checked 20.8.2006

[17] M. Selhorst and C. Stueble *Linux TPM driver*, website: *http://www.prosec.rub.de/tpm/* , checked 20.8.2006

[18]KaZaa,website:http://www.kazaa.com/us/help/glossary/ participation_ratio.htm , checked 20.8.2006

[19] Edward Felten, Understanding Trusted Computing: Will its benefits outweight its drawbacks?, IEEE Security and Privacy 1, 2003

[20] J. Poritz and C. Cachin, *Trust[ed | in] computing, signed code and the heat death of the interne*, Symposium on Applied Computing, TRECK track, 2006

[21] R. Sandhu and X. Zhang, *Peer-to-Peer Access Control Architecture Using Trusted Computing Technology*, Symposium on Access Control Models and Mechanisms 2005

[22] A. Sadeghi and C. Stuble. *Taming trusted platforms by operating system design*, Information Security Applications, 4th International Workshop, LNCS 2908, 2003

[23] E. Brickell, J. Camenisch, and L.Chen, *Direct Anonymous Attesation*, Proc. Of ACM CCS, 2004

[24] Trusted Computing Group, *Trusted Network Connect Architecture*, Version 1.1 , Standard Specification, 2006

[25] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, *The EigenTrust Algorithm for Reputation Management in P2P Networks*, vol. Proceedings of the Twelfth International World Wide Web Conference, 2003.

[26]S. Song, K. Hwang, R. Zhou, and Y.-K. Kwok, *Trusted P2P Transactions with Fuzzy Reputation Aggregation*, IEEE Internet Computing Magazine, 2005.

[27]E. Damiani, S. D. C. d. Vimercati, S. Paraboschi, and P. Samarati, *Managing and sharing servants' reputations in P2P systems*, vol. Transactions on Knowledge and Data Engineering: IEEE, 2003, pp. 840-854.

[28]S. Marsh, *Formalising Trust as a Computational Concept*, Department of Mathematics and Computer Science, University of Stirling, PhD Thesis 1994.

[29]J.-M. Seigneur, *Trust, Security and Privacy in Global Computing*, Trinity College Dublin, PhD Thesis Technical Report TCD-CS-2006-02, 2005.

[30]J. R. Douceur, *The Sybil Attack*, vol. Proceedings of the 1st International Workshop on Peer-to-Peer Systems, 2002.

[31] C.-N. Ziegler and G. Lausen, *Spreading Activation Models for Trust Propagation*, presented at the International Conference on e-Technology, e-Commerce, and e-Service, 2004.