Trusted Computing: Providing Security for Peer-to-Peer Networks

Shane Balfe, Amit D. Lakhani and Kenneth G. Paterson,
Information Security Group,
Royal Holloway, University of London,
United Kingdom.
{S.Balfe,Amit.Lakhani,Kenny.Paterson}@rhul.ac.uk

Abstract

In this paper, we demonstrate the application of Trusted Computing to securing Peer-to-Peer (P2P) networks. We identify a central challenge in providing many of the security services within these networks, namely the absence of stable verifiable peer identities. We employ the functionalities provided by Trusted Computing technology to establish a pseudonymous authentication scheme for peers and extend this scheme to build secure channels between peers for future communications. In support of our work, we illustrate how commands from the Trusted Computing Group (TCG) specifications can be used to implement our approach in P2P networks.

1. Introduction

It seems likely that computing platforms compliant with Trusted Computing Group (TCG) specifications will become widespread over the next few years. This is evident from initiatives such as Intel's LaGrande technology[1] and Microsoft's Next-Generation Secure Computing Base(NGSCB)[2], which are aimed at developing core hardware architecture and operating system components based on TCG specifications respectively. Once one accepts that the Trusted Computing paradigm offers an interesting and powerful set of security features, the natural question arises: for what purposes can this technology be exploited? In this paper, we examine the application of Trusted Computing to securing Peer-to-Peer (P2P) networks.

Security issues within P2P networks have not yet been widely addressed. A major conflict arises from the perceived requirement to provide anonymity for users of P2P networks and an increasing need to provide robust access control, data integrity, confidentiality and accountability services. Pseudospoofing attacks, in which malicious parties are able to claim multiple identities, plague P2P networks, and represent a fundamental security threat. For without the foundation of stable, verifiable identities, it is difficult to provide a set of desired security services. These are increasingly important as industry moves towards using P2P technology in applications and as P2P e-commerce emerges. The security situation for P2P networks is made worse because, by definition, they lack any centralized authority who can vouch for identities or security parameters.

In this paper, we demonstrate how features of the TCG specifications [8, 9] can be employed to enhance the security of P2P environments. In particular, we show how the TCG protocols for Direct Anonymous Attestation (DAA) can be used to enforce the use of stable, platform-dependent pseudonyms and thus reduce pseudospoofing in P2P networks. Taking this a step further, we show how runs of DAA protocol can be used to build entity authentication at the level of pseudonyms and can be securely linked to the establishment of secure channels with known endpoints. While earlier authors [7, 14] have posited the potential application of Trusted Computing in P2P networks, we do not halt at that point. Rather, we go as far as describing the specific TCG mechanisms and commands which enable us to establish security in P2P networks. Another factor that differentiates our approach is the degree of TCG specification functionality that we use. Our proposal does not require integration with trusted operating system components based on TCG specifications, nor is it prescribed for a particular hardware architecture. Rather we use the functionalities as provided in current TCG-enabled platforms available in

1 http://www.intel.com/technology/security/

2 http://www.microsoft.com/resources/ngscb/default.mspx

the market today.

The paper starts with identifying the major security issues in P2P networks in Section 2. Having identified these issues, we give a short overview of Trusted Computing technology in Section 3, detailing the key features we use in our approach. Section 4 describes how Trusted Computing can be used to build a pseudonymous authentication scheme within P2P networks, and to build secure channels between participating peers. We conclude the paper by listing some open issues and conclusions in Section 5 and 6.

2. Security in P2P networks

Whilst many aspects of P2P networks have been thoroughly researched, security within these networks still remains a challenge. It is not our intention to exhaustively review security issues for P2P networks here. Indeed, good overviews of this topic can be found in [4] and [15]. Rather, we focus on identifying some key security issues in P2P networks.

The security architecture [12] associated with the ISO/ITU Open Systems Interconnection (OSI) reference model serves as a useful framework for assessing security issues in networks. According to [12], a secure system is governed by the set of security services it provides, and the mechanisms put in place to cater for these services. The set of potential security services in [12] are divided into five main classes:

- Confidentiality,
- Integrity,
- Authentication – including data origin authentication and entity authentication,
- Access Control, and
- Non-repudiation.

Additional services not explicitly included in [12] could be added to this list. Accountability and anonymity are foremost candidates if we focus on P2P networks. Any particular network may require a combination of all, some, or even none of these various security services.

In traditional client-server systems, users are typically identified with a user account, and system-specific controls can be mounted on these accounts to provide security mechanisms. Security services such as access control and accountability can be implemented in this manner, with the accounts providing a form of stable identity. Other services such as authentication and non-repudiation clearly also rely on the establishment and preservation of stable identities. Thus most (though not all) of the generic security services are reliant on the provision of stable identities.

In a P2P environment, there are by definition no trusted third parties who can provide assurances as to the identities of entities in the network. Instead, each peer is typically identified based on a *handle* or *pseudonym* that is selected by the peer for itself. In most P2P networks, there exists no standard registration procedures on these pseudonyms, and indeed entities can claim multiple pseudonyms, including those of other entities in the network. Pseudospoofing, a term coined by Detweiler [5], refers to a peer creating and handling more than one pseudonym at once. Potentially, an attacker might manipulate tens or even hundreds of pseudonyms to his advantage.

An attacker might also make use of the pseudonym of an existing peer, preferably one with a good reputation. This type of attack has been named *ID stealth* in the literature [3], but we prefer the name *pseudotheft*. An attacker who engages in pseudospoofing or pseudotheft in a P2P network may do so in order to manipulate a reputation mechanism in place in that network. A peer, having gained a bad reputation, can discard the corresponding pseudonym and re-join the network with a fresh pseudonym. In on-line auction systems, a peer can use a pseudonym to generate false bids, an attack known as *shilling*. In general, a peer can create a number of pseudonyms, build up the reputation of one or more of their pseudonyms using the others, and then make use of the now reputable pseudonym(s) to persuade other peers to trust him. Clearly any P2P network attempting to rely on a reputation system for building trust in peers would need to address the problems of pseudospoofing and pseudotheft. As an example, the reputation system used in eBay, essentially a P2P system, has constantly been under pseudospoofing attacks[3]. It has been argued [6] that in any P2P system without a centralised point of trust, such attacks on identity are endemic and can never be effectively combatted.

Other security services like access control, which in P2P networks mean restricting access to authorised peers, also rely heavily on authentication of peers. The (deliberate) absence of such robust authentication schemes have led to blatant abuse of copyright laws and the consequent litigations against P2P-software companies. It is also evident that pseudotheft can lead to loss of confidentiality. With the emergence of P2P e-commerce, confidentiality of communications and integrity of data in transit will play a significant role, but with no stable identities and authentication mech-

3 An interesting case, "The Saratoga Fakes", has been compiled by the Stamp Collectors Against Dodgy Sellers (SCADS) institute, `http://www.scads.org/alterations/Saratoga.htm`

anisms in place, secure communications in such transactions over open networks are at considerable risk.

The approach of using pseudonyms provides a degree of anonymity for peers and eases discovery of resources and routing of resource queries in P2P networks, but is clearly in conflict with the requirement for stable identities that is needed, as we have noted above, to provide many security services. This problem of providing entity authentication in the face of untrusted networks of peers seems to us to be a central challenge in providing wider security services for P2P networks. In the next section, we try to address this challenge with the help of TCG functionality.

3. Overview of Trusted Computing

3.1. Background

Before we move on to the focus of our work it is necessary to introduce certain core aspects of TCG technology. In particular, we review the mechanisms involved in integrity measurement and storage. The rationale behind measuring and storing integrity altering events is that it allows a platform to transition into any number of possible states which can be made visible to interested external parties, but the platform is unable to falsely adduce its current state.

At the root of the TCG specifications is a tamper-resistant security hardware device, which is embedded in the platform, called the *Trusted Platform Module* (TPM). The TPM acts as a "root of trust" for the platform and has integrated cryptographic functionality providing integrity, creation and use of digital signatures and privacy protecting mechanisms. A Platform Configuration Register (PCR) is a 20-byte storage area, internal to the TPM, which contains a cumulative digest of a number of measured values, typically consisting of embedded data. The specification mandates that there be at least sixteen PCRs within a TPM. The PCR update mechanism is of particular importance to our approach and is termed as *extending the PCR*. It is expressed programmatically as:

$$\text{PCR}[n] \leftarrow \text{SHA-1}(\text{PCR}[n] \,\|\, \text{measured data}).$$

Here n is the number of the PCR being updated. Another structure that is relevant for integrity measurement is the Stored Measurement Log (SML), alternatively refered to as the Event Log, which is responsible for maintaining an ordered database of integrity changing events. The SML and PCRs are linked by the fact that individual PCRs store digests of measured values while the SML acts as an organised store for all the events for every PCR. As it is necessary to store the complete event history for all the PCRs, the SML can potentially grow quite large. For this reason the SML may be stored outside the TPM in a non-shielded location. The data held in the PCR, alongwith the relevant portion of the SML, is used as evidence to attest to the current platform state.

The PCRs and SML are intimately related. Without the representative digests contained in PCRs, the events maintained by the SML are meaningless (as there is nothing to stop someone from generating false events in the log). Correspondingly, without the relevant portion of the SML, a PCR digest is devoid of any meaning (as the context from which the digest was generated would be lost). Consistency between the two structures is preserved by the fact that PCRs are maintained internally to the TPM and that PCR extension operations are only permitted via TPM protected capabilities. These PCRs provide evidence of attempted tampering of the log, since the sequence of events tied to a specific PCR can be extracted from the SML, rehashed and compared to the actual value contained in the PCR.

Now that we have a basic understanding of how integrity measurements are taken and stored within a TCG-compliant platform, we can move on to examine the concept of integrity reporting and attestation.

3.2. Platform Attestation

When a verifier wishes to inspect a host platform's configuration he may request a specific set of PCR values. An assurance as to the validity of a PCR value is provided by means of a signature on the register value produced by the TPM. The TPM uses the private component of a key pair called an Attestation Identity Key (AIK) to perform the signing operation. Within a TCG conformant platform, AIK key pairs are used as aliases for another key pair called the Endorsement Key (EK). There is no prescribed limit on the number of AIKs that can be used within a platform. This provides an anonymity mechanism, whereby the TPM can use different AIKs each time it signs a PCR value, making those attestations unlinkable. The reason for introducing this anonymity mechanism is that the EK uniquely identifies the platform, and possibly the owner of the platform. This is the basic reason why the TCG specifications essentially mandate that the EK should never leave or be exposed outside the TPM.

The actual TCG attestation protocol is outlined below:

1. A Challenger indicates that it wishes to inspect one or more PCR values from a platform.

2. A Platform Agent gathers the relevant SML entries corresponding to the PCR values requested.

3. The TPM sends the Platform Agent the requested PCR values.

4. The TPM attests to the values contained in the requested PCRs by producing a signature using an AIK private key.

5. The Platform Agent assembles credentials that vouch for the TPM. The signed PCR values with their corresponding SML entries and relevant credentials are returned to the Challenger.

6. The Challenger verifies the supplied data. The measurement digest is computed from the returned SML entries and compared with the signed PCR values. The platform credentials are evaluated and signatures checked.

As we can see, a Challenger obtains a portion of the SML (possibly all of it) and one or more PCR values allowing it to confirm that the platform is indeed in a specific state by examining sequences of events. The way in which a platform proves that it is in possession of the relevant credentials differs in versions 1.1 and 1.2 of the TCG specification. In version 1.1, it was envisaged that the platform would obtain a credential from a trusted third party known as a Privacy CA. The credential would come in the form of digitial certificate issued by the Privacy CA on a specific AIK, with the Privacy CA being aware of the binding between AIK and EK. Thus the privacy properties of the scheme would depend on the security, reliability and trustworthiness of the Privacy CA. Version 1.2 still supports this Privacy CA mechanism, but also introduces a new approach called Direct Anonymous Attestation (DAA). DAA, the subject of the following section, uses more sophisticated cryptographic techniques to ensure the privacy of users, without introducing the requirement for special trusted third parties.

3.3. Direct Anonymous Attestation

We provide here a brief overview of the key features of DAA that we will use in building security for P2P networks. The full cryptographic details can be found in [2], while the TCG specification [9] contains implementation specifics.

Before a platform can provide attestations to a Challenger (called a *verifier* in [2]) concerning its current configuration state, it is necessary for the platform to go through a `join` phase where it obtains a credential from a trusted third party called an *issuer*. In practice, the part of the issuer might be played by the platform manufacturer or by a third party who wishes to offer services exploiting TCG features.

During the `join` phase the platform is asked to become a member of the group of platforms to which credentials have been issued by that issuer. During this process, the platform proves that it has knowledge of a specific TPM-controlled, non-migratable secret value f and is authenticated to the issuer via its EK. If the issuer accepts, then it provides to the platform a credential in the form of a *Camenisch-Lysyanskaya (CL) signature* [1] on the secret value f. It is this CL signature that will later enable the platform to convince a verifier that it has previously successfully completed the `join` phase with a particular issuer. Note that the issuer does not actually learn f in this process.

In order for a platform to obtain a credential for its secret value f, the platform must reveal a pseudonym N_I of the form ζ_I^f, where ζ_I is an element of some suitable group that is derived from the issuer's name. By maintaining a history of pseudonyms that it has seen, the issuer can check if credentials have already been issued to this TPM. The issuer can also check for rogue platforms at this stage, by testing if $N_I = \zeta_I^f$ corresponds to an f appearing on a blacklist.

Once the `join` phase is complete, the TPM (with the help of the platform) can use the `DAA-Signing` algorithm to prove to any verifier that it has an appropriate issuer credential (in the form of a CL signature), and at the same time, sign a message m using the secret value f. Here, m is typically the public component of an AIK key pair, but, according to [2, Section 4.6], it can also be an arbitrary string. In the former case, the output of the `DAA-Signing` algorithm amounts to an attestation that the TPM is in possession of a valid issuer credential and has a particular AIK. This gives the verifier a similar assurance concerning that AIK's linkage to a particular platform to that gained using the Privacy CA approach described in the previous section. In turn, this allows any PCR value signed by the particular AIK to be checked as being genuine by the verifier. Corresponding to the `DAA-Signing` algorithm is a `DAA-Verify` procedure that is executed by the verifier.

In the `DAA-Signing` algorithm, the platform is only identified by a pseudonym. As in the `join` phase, this pseudonym is of the form $N_V = \zeta^f$, where now the selection of ζ determines the anonymity properties of the attestation process. The value of the base ζ will typically be chosen by the verifier. In this case, if the verifier fixes on a choice of ζ (deriving ζ from its name, for example), then this will also fix the value of the pseudonym N_V used by a particular platform. This will allow all the transactions involving a particular TPM to be linked. It is worth pointing out here that the secret value f is non-migratable, so an f value should not be able to leave the confines of a TPM, and hence only the TPM that generated that value should be capable of using it. The value of ζ might also be selected at random by the platform for each transaction with the verifier – this would allow the platform's transactions to become completely unlinkable.

4. Building Security Services in P2P Networks from Trusted Computing

In this section, we consider how the use of stable pseudonyms can be enforced in any P2P network in which each peer is TCG-enabled and has appropriate DAA credentials.

The mechanism for enforcing stable pseudonyms is simple. We assume only that the P2P network has a particular name, that is, a unique identifier. The pseudonyms used in our network are strings of the form $N_P = \zeta^{f_P}$, where ζ is derived from the network name by applying a suitable hash function to produce a group element and f_P is the secret associated with the TPM located on the peer P claiming the pseudonym.

Whenever a peer claims a particular pseudonym, the peer verifying that claim can challenge the claimant to supply a DAA signature on, say, a random message M. We can also exploit this property to enable other security services by selecting M to include additional values. The verifying peer can check, using the `DAA-Verify` algorithm, that the claimant has a valid credential supplied by a particular issuer and can be convinced that the value f_P used in forming the claimed pseudonym N_P is the same one claimed at the time of credential issuance. The use of a random message M prevents replay attacks in which a rogue peer captures and replays credentials. Through these checks, the pseudonym N_P that can be claimed by peer P is fixed and determined as a function of the network name and the particular f_P certified during the `join` phase. In summary, we have built an *pseudonymous authentication mechanism* from the DAA algorithms. Through this mechanism, peers can be authenticated by other peers, but platform identities are not revealed in the process.

To make this a workable method for enforcing stable pseudonyms, we need to ensure that a particular platform will only ever obtain one credential from one of a set of authorised issuers that are recognised by the verifying peer. Otherwise, it may be difficult to prevent a peer obtaining multiple credentials for different f values, from one or more issuers, and using these to produce multiple pseudonyms. One mechanism for ensuring that a platform can only ever obtain a single credential is to assume that an initial credential is issued to the platform in a controlled fashion at the time of manufacture. The set of authorised issuers is then the small set of platform manufacturers and the peer software can be configured with their public keys and to insist on seeing a credential issued by one of these manufacturers. Here, then, the manufacturers become the root of trust for DAA. This is expected to be a likely scenario in the deployment of trusted computing. However, having a trusted service provider as a root of trust is a viable alternative that is expanded upon in the extended version of this work. We merely present the manufacturer as a root of trust for ease of explanation.

To what extent does our approach prevent pseudospoofing and pseudotheft attacks? We note that, if our condition on credential issuance is met, then a peer cannot claim multiple different pseudonyms. Nor can a peer claim the pseudonym of another peer. Thus these attacks, so destructive in P2P networks, are prevented. The usual blacklisting mechanisms can be used by peers to detect the use of a rogue TPM, provided the relevant information can be distributed and kept up-to-date. Of course, nothing in our approach prevents an attacker from purchasing multiple TCG-compliant platforms and using these to create multiple pseudonyms. However, this kind of attack implies an economic cost for the attacker.

It is clear that, by enforcing the use of pesudonyms, a given peer's actions on the network do become linkable, so our approach does not offer a full level of anonymity. However, the use of DAA never reveals the platform identity (in the form of the endorsement key) to verifying parties so the peer actions cannot be linked to a particular TPM. Notice too that the pseudonym used during the `join` phase is different from that revealed during `DAA-Signing` (because $\zeta_I \neq \zeta$ in general). Thus the pseudonyms do shield the actions of a peer effectively . We also note that the pseudonyms we propose for use do not relate to particular individuals; rather they are linked to particular platforms (more precisely, TPMs).

A given peer may wish to operate in more than one P2P network. Because of the use of the network name in determining the value of ζ used to form pseudonyms, we obtain the nice feature that a peer's actions in different networks will remain unlinkable.

Finally in this section, we note that a somewhat less attractive authentication mechanism can be built using the legacy Privacy CA approach. The idea is to extend PCRs using nonces exchanged between peers as measured data. The PCRs can be signed using AIKs, which are in turn signed by the Privacy CA. The TCG attestation protocol then provides a mechanism to exchange signed nonces. We omit the details here, as this approach is closely related to the technique introduced in the next section.

Intuitively, we can also extend this authentication scheme to support access control. In the simplest form, on successful pseudonym verification, a peer can use access control lists to grant or disallow access to specific peer(s) based on a local or network-specific access control policy. Clearly a more sophisticated and application-dependent access control system can be implemented on top of these stable pseudonyms.

4.1. Building Secure Channels between Peers in P2P networks using Trusted Computing

Whilst authentication (even at the level of pseudonyms) is a very useful service to provide in a P2P network, its usefulness is limited if it cannot be extended to provide protection for an entire communication session between peers. We address this next, showing how our authentication mechanism can be extended to an authenticated key establishment protocol. This in turn allows the provision of secure tunnels between peers, enabling confidentiality and integrity services for data in transit.

Suppose we have two peers P and Q, with pseudonyms N_P and N_Q, in a particular P2P network. We assume these peers wish to authenticate one another and establish keying material for protecting further communications.[4]

In the simplest form of our approach, the peers exchange random nonces R_P, R_Q, ephemeral Diffie-Hellman values g^{x_P}, g^{x_Q} (where g is a generator of a suitable group negotiated in advance or by peer software configuration), and signatures on the 160-bit hash[5] of the message:

$$M = N_P \| N_Q \| R_P \| R_Q \| g^{x_P} \| g^{x_Q}.$$

The signatures are obtained by each peer using their TPM and platform to execute the DAA-Signing algorithm on the hash of M. Using the DAA-Verify algorithm, each peer can check that the other has a valid credential supplied by a particular issuer and can be convinced that the values f_P, f_Q used in forming the claimed pseudonyms N_P, N_Q are the same as those claimed at the time of credential issuance. Here, we are essentially re-using the pseudonymous authentication mechanism from Section 4, but with M made up of a longer string including pseudonyms, nonces and Diffie-Hellman values. If both signatures are confirmed by DAA-Verify, then the parties can safely compute common secret keying material $K = g^{x_P x_Q}$, and then derive keys for MAC and symmetric encryption algorithms from K.

Here, we have used the facility that a platform can ask the TPM for the DAA signature on any message M. However, if we look at the more restrictive case of only allowing DAA-signing on AIKs, we can use an alternative (but slightly less efficient) approach involving PCRs and AIKs. Each peer can use the string $M = N_P \| N_Q \| R_P \| R_Q \| g^{x_P} \| g^{x_Q}$ as measured data to extend a PCR, using the process outlined in Section 3. Then each peer can request its TPM to produce an attestation to the PCR value, which takes the form of a signature on the PCR value produced using the private component of the peer's AIK. Next, at each peer, the DAA-Signing algorithm can be used to sign the peer's public AIK component. Finally, the peers can exchange the DAA signatures on their AIKs and the AIK signatures on the string M. Checking these signatures provides the necessary authentication for the Diffie-Hellman key exchange. This approach can be realised in practice by first extending the PCRs and then interleaving two runs of the TCG attestation protocol outlined in Section 3.2.

In the full version of this paper we show how these methods for performing authenticated key exchanges can be integrated with the SSL/TLS and IPSec protocol suites.

4.2. Implementation

In order to make our proposals for TCG-based authentication and secure channel establishment mechanisms as concrete as possible, we give here an indication of the specific TCG commands that are needed to implement our ideas.

In order to understand the mechanisms for extending a PCR register we need to examine the event logging mechanism for trusted computing in greater detail. We will base our discussions on the assumption that our P2P application is running on a TCG-conformant platform. The general principles and commands discussed here are transferable to any TCG-enabled device. As we mentioned previously, the SML is not really a log. It is better to envisage it as an array of events in which each entry is in the form of

4 The authentication need not be mutual, and the alterations to our methods needed in this case are trivial.

5 The TCG specification limits externally generated messages that are to be DAA signed to be at most 160 bits in length.

a TSS_PCR_EVENT structure [8, p.3]. In a TCG-enabled device it is a TSS_PCR_EVENT that provides information regarding individual PCR extension events. The rgbEvent parameter in a TSS_PCR_EVENT is the one we are interested in, as it is a pointer to event information data, in our case a string formed from cryptographic parameters. It is the TCS (TCG Core Services) Event Log services that are responsible for maintaining the SML. These services are responsible for allowing PCR extension events to be logged and allowing challengers interested in these events access to them.

To incorporate an event into a PCR two things must happen. A call to the TPM_Extend command [11, p.146] is needed. This is the command that is responsible for extending the PCR. Then an additional call to a function such as Tcsi_LogPcrEvent is needed. This function is responsible for adding the new event to the end of the array associated with a named PCR [8, p.230]. When a challenger wishes to verify a system by examining one or more PCRs (as indicated in step 1 of the TCG attestation protocol) the platform must perform a number of operations to satisfy this request. A call to Tcsi_GetPcrEventsByPcr returns an event log of all events in the SML that are bound to a single PCR [8, p.234]; this can be called multiple times depending on the number of requested PCR values. The event log is returned as an ordered sequence of TSS_PCR_EVENT structures. Attestation of the chosen PCR values occurs after a call to TPM_Quote [11, p.148]. This operation is responsible for providing cryptographic reporting of PCR values, using an AIK key to sign the current value of a chosen PCR.

The TPM_DAA_JOIN command [10, p.136] is responsible for obtaining the issuer's CL signature on the DAA secret f, while the TPM_DAA_SIGN command [10, p.138] is responsible for running the DAA-Signing algorithm and, through this, proving that the TPM in question does indeed possess valid credentials. By performing the TPM_DAA_SIGN command upon the AIK key used in the TPM_Quote command, a TPM can prove to the verifier that it is in possession of the private component of the AIK key that was used to sign a requested PCR value.

5. Issues and Open Problems

In this section, we briefly summarise the limitations of our approach and topics for further research.

One of the foremost issues is that of data representation. There is something of a semantic gap between dealing with the sequence of ones and zeros that is a DAA pseudonym and working with the more traditional 'handles' to which users of P2P networks are accustomed. One possible solution to this problem is to use a linked local name space with a network name. If a peer on a particular network maintains a list of pairs (ζ, ζ^f) then with each of these pairs, it can associate a particular name or handle. In this scenario ζ represents the network and ζ^f a particular pseudonym on that network. This mapping would be akin to that used in SPKI/SDSI [13] but instead of using a public key for identification as in SPKI/SDSI, we would use (ζ, ζ^f). This kind of mechanism is suitable for use in a P2P network as it is not necessary for peers to know every other peer on a network. Instead, they need only know those peers with whom they have previously interacted.

Another area requiring further development is the building of reputation models based on these stable pseudonyms. This should be relatively straightforward given the work we have done. As an example of just one approach, peers could provide DAA-signed references for other peers, based on successful interactions. A peer in possession of a number of references could display these to third parties to establish a reputation. It would be interesting to see various rating schemes examined in a TCG-enabled P2P setting. In addition, the ability to transfer a peer's pre-existing reputation from one network on which it is known to a new network would be another potential path for future work.

Credential replacement is also a non-trivial task in DAA. If it is necessary to issue replacement credentials for a TPM then the previous DAA history of the platform gets erased, at least as far as the DAA credentials from that issuer are concerned. This potentially allows a rogue TPM to rejoin a network from which it has previously been barred. However decisions on credential replacement are a matter of policy for a given issuer and are application-dependent. Credential reissue is less of a problem as (typically) a rerun of the Join phase with the same issuer will result in the same credentials being reissued to a platform.

In the event of a catastrophic hardware failure that results in the TPM no longer working, all data internal to the TPM as well as all external data controlled by the Storage Root Key (SRK) will be lost. In order to address this problem the TCG specifications detail an optional (vendor-specific) maintenance capability [10, p.73] that allows a platform owner to create a backup of all data held in protected storage within a TPM (including migratory and non-migratory data). If a manufacturer chooses to implement this feature it injects a public key into the TPM's non-volatile storage area at the time of chip manufacture. This key is then used when the platform owner runs the TPM_CreateMaintenanceArchive [11, p.103] command which creates a maintenance blob encrypted with the manufacturer's public key. This blob can only be decrypted by the manufacturer and can (under certain tightly controlled circumstances) be initialised into a new TPM subsystem. In this way any reputation or goodwill that was built up by our pseudonym within the network may be preserved.

6. Conclusions and Future Work

Many security services within networks depend on the establishment of stable identities of network users. In this paper, we examined the application of Trusted Computing in establishing such stable identities within P2P networks. Our approach relies only on TCG functions that are available in implementations today rather than on proposed mechanisms such as curtained memory, integration with secure operating systems based on TCG specifications or compartmentalised applications that may be available in the future. In the extended version of this paper, we present models and application scenarios where secure communications between peers using TCG-enabled devices is envisaged. One perhaps counter-intuitive implication of our work is that TCG-enabled P2P networks can make P2P networks harder to effectively police. On the other hand, our work also reflects a possible way forward in using TCG to enable secure P2P e-commerce, through the development of secured commercial P2P networks and of P2P networks free from pseudospoofing and pseudotheft.

7. Acknowledgements

We wish to convey our thanks to Graeme Proudler and Liqun Chen of Hewlett-Packard Laboratories, Bristol for answering many questions on the TCG specification and its application.

References

[1] J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *Security in Communication Networks, Third International Conference, SCN 2002*, volume 2576 of *LNCS*, pages 268–289. Springer Verlag, 2003.

[2] L. Chen, E. Brickell, and J. Camenisch. Direct anonymous attestation. In V. Atluri, B. Pfitzmann, and P. McDaniel, editors, *Proceedings of 11th ACM Conference on Computer and Communications Security(CCS'04)*, October 2004.

[3] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In V. Atluri, editor, *Proceedings of the 9th ACM conference on Computer and Communications Security*, pages 207–216. ACM Press, 2002. ISBN 1-58113-612-9.

[4] N. Daswani, H. Garcia-Molina, and B. Yang. Open problems in data-sharing peer-to-peer systems. In D. Calvanese, M. Lenzerini, and R. Motwani, editors, *Proceedings of 9th International Conference on Database Theory (ICDT'03)*, volume 2572 of *LNCS*, pages 1–15. Springer, 2003.

[5] L. Detweiler. The snakes of medusa – internet identity subversion, 1993. Cypherpunks mailing lists.

[6] J.R. Douceur. The sybil attack. In P. Druschel, M. F. Kaashoek, and A. I. T. Rowstron, editors, *Peer-to-Peer systems, First International Workshop, IPTPS 2002*, volume 2429 of *LNCS*, pages 251–256. Springer, 2002.

[7] T. Garfinkel, M. Rosenblum, and D. Boneh. Flexible OS support and applications for Trusted Computing. Preprint.

[8] Trusted Computing Group. *TCG Software Stack Specificiation Version 1.1*, 2003. `https://www.trustedcomputinggroup.org/downloads/specifications`.

[9] Trusted Computing Group. *TCG Specification Architecture Overview Revision 1.2*, 2004. `https://www.trustedcomputinggroup.org/downloads/specifications`.

[10] Trusted Computing Group. *TPM Main: Part 1 Design Principles*, 1.2 edition, 2005. `https://www.trustedcomputinggroup.org/downloads/specifications`.

[11] Trusted Computing Group. *TPM Main: Part 3 Commands*, 1.2 edition, 2005. `https://www.trustedcomputinggroup.org/downloads/specifications`.

[12] International Organisation for Standardization. *ITU-T Rec. X.800—ISO 7498-2, Information processing systems-open systems interconnection - basic reference model - part 2: security architecture.* ISO/ITU, 7498-2 edition, 1989.

[13] R. Rivest and B. Lampson. SDSI: a simple distributed security infrastructure, 1996. presented at CRYPTO'96 Rumpsession.

[14] S. E. Schechter, R. A. Greenstadt, and M. D. Smith. Trusted computing, peer-to-peer distribution and the economics of pirated entertainment. In *The Second Workshop on Economics and Information Security.* May 2003.

[15] D. S. Wallach. A survey of peer-to-peer security issues. In M. Okada, B. C. Pierce, A. Scedrov, H. Tokuda, and A. Yonezawa, editors, *Software Security – Theories and Systems, International Symposium, ISSS 2002*, volume 2609 of *LNCS*, pages 42–57. Springer, 2003. ISBN 3-540-00708-3.