Trust in Peer-to-Peer Content Distribution Protocols

Nicolai Kuntze, Carsten Rudolph, Andreas Fuchs

▶ **To cite this version:**

Nicolai Kuntze, Carsten Rudolph, Andreas Fuchs. Trust in Peer-to-Peer Content Distribution Protocols. 4th IFIP WG 11.2 International Workshop on Information Security Theory and Practices: Security and Privacy of Pervasive Systems and Smart Devices (WISTP), Apr 2010, Passau, Germany. pp.76-89, 10.1007/978-3-642-12368-9_6 . hal-01056065

HAL Id: hal-01056065
https://inria.hal.science/hal-01056065

Submitted on 14 Aug 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Trust in Peer-to-Peer content distribution protocols

Nicolai Kuntze, Carsten Rudolph, Andreas Fuchs

Fraunhofer Institute for Secure Information Technology (SIT)
Rheinstrasse 75, 64295 Darmstadt, Germany
{nicolai.kuntze|carsten.rudolph|andreas.fuchs}@sit.fraunhofer.de

Abstract. The distribution of virtual goods like multimedia data relies on the trustworthiness of the distribution system. Recent concepts for the distribution based on peer to peer networks like BitTorrent require new approaches to establish the needed level of trust in the overall functionality of the system.
This paper explores the integration of hardware based trust concepts from the domain of Trusted Computing in the well know BitTorrent protocol suite.

1 Introduction

The commercial distribution of virtual goods in the Internet has relied on a client server model for the last years. A central entity provides the goods and clients just receive the content. Various systems like the well known iTunes are based on this paradigm. Aside this traditional approach of service delivery distributed solutions arose like distributed caching, Content Distribution Networks (CDNs) [7], and more recently peer-to-peer (P2P) networks as for example BitTorrent [14]. However, the (legal) use of these more efficient distribution protocols has so far been mostly applied to freely available data, like open source software. Nevertheless, there are good reasons motivating the use of efficient peer-to-peer distribution protocols also for commercial applications. First industrial applications based on the BitTorrent protocol were already available as for example BitTorrent Inc. rental service and Joost [3]. As BitTorrent Inc. used the standard protocol suite but used a DRM (Microsoft), Joost used a close client software together with secure coding and other additional obfuscation technologies.

Today's cost structure of the infrastructure operated by Internet Service Providers (ISPs) is not optimal for traditional client server distribution models. Obviously, the bandwidth on the side of the server, i.e. on the side of the backbone infrastructure is heavily used, while highly ex-pensive as its utilization is already high. On the other side it is recognized that the part of the infrastructure physically near to the end user, sometimes referred as last mile, often has unused bandwidth and also disk space and computing power. Therefore it is desirable from the point of view of the ISPs to push traffic associated with the content distribution to a certain extend into this part of the infrastructure and

also to reduce the traffic between the peers. This results in distribution networks based on nodes located at the edges of the ISP networks, e.g. situated in the households of the end users.

Such a network design allows for optimized bandwidth utilization in the core network by applying optimized P2P protocols that take the physical configuration of the ISP network into account and offers proper caching and data propagation methods. Additionally costs for expensive central server systems can be reduced as the complexity of these systems is diminished due to the abilities inherent to a P2P network like redundancy and load balancing. Also of interest is an improved Quality of Experience (QoE) considering a video on demand (VoD) use case. There, the network is able to store the data expected to be viewed on the device of the end user or in his neighborhood. Such functionality is based on certain consumption data known to and analyzed by the overall system operated by the ISP or by the application provider.

However, existing peer-to-peer distribution protocols currently don't have security build in. Thus, deployment in commercial applications is not straightforward. As the ISP is not the owner of the virtual goods in most of the cases his aim is to provide an interesting platform for the actual owners of the content and to allow each separate Content Provider to offer and protect their content according to their needs. It is important to see here that the goods offered may be very different in their needs as e.g. an online game has different requirements then a VoD service. Therefore different (concurrent) offerings are hosted in the same node at the same time. Sharing of one environment between different customers is already a well known business case like Web based storage services, Web Email, or resizable computing capacity as offered e.g. by Amazon.

The EU research project Nanodatacenters (NaDa) develops a platform that provides a basic set of functionalities to establish a trustworthy environment at the side of the user households that can be used by the Content Providers to offer their goods. One part of this system is the development of secure and trustworthy peer to peer protocols. This paper presents a concept for the establishment of a trustworthy P2P system fit for the presented operation model exemplified as a protocol extension of the well known BitTorrent file sharing protocol.

In the context of ISP based operation BitTorrent provides the benefit of the tracker based operation as here the operator has some impact on the discovery of the respective data sources.

The BitTorrent protocol [4] distributes data over a large number of nodes of a P2P-network. A so-called tracker is then used to locate nodes that provide requested files. A meta file, so called torrent file, provided by the initial data provider contains initial data on the content, like size in terms of chunks and hash values. BitTorrent distinguishes between the meta file distribution, tracker protocol for providing meta information to peers, and the peer wire protocol for transfer of the actual content. Using the given data from the meta data file the tracker is then contacted using the tracker protocol. After receiving the list of nodes, some of these nodes are contacted using the peer wire protocol and

requesting certain pieces of data. All nodes downloading and uploading parts of the same content are called a swarm.

By controlling the node lists provided by the tracker to the requesting nodes the ISP can steer the traffic and avoid inter e.g. ISP traffic [5] but also manage the usage of his backbone infrastructure. BitTorrent is also a well known factor to the ISP infrastructure.

It is essential to note that each node participating in a BitTorrent swarm first was connected to a tracker and keeps the connection to the tracker alive throughout operation. BitTorrent initially started as a centralised tracker protocol providing exact one tracker for each torrent. Due to various reasons the protocol was improved and a distributed tracker approach was introduced using Kademlia [12] as the underlying routing approach.

Such an, also distributed, architecture can also be used to establish the management of the swarm. Each node registers itself to at least one tracker who also serves as a distribution point for network wide management messages. Through this network each node can be addresses and also additional support protocols for e.g. routing of messages in an ISP friendly [10] way. This approach only relies on the tracker protocol.

The rest of the paper is organised as follows. In Section 2 a security analysis of the BitTorrent protocol in the NaDa use case is presented. Section 3 revises Trusted Computing as it is used for the security enhanced protocol. Section 4 introduces briefly the notation that is used to present the enhanced protocol in Section 5. After the protocol presentation a short security evaluation is presented in section 6. The paper concludes in Section 7.

2 Security analysis

BitTorrent currently totally depends on the honesty of peers. As there is no security build in it does not come as a surprise that a variety of attacks is possible. Attacks include those presented in [16] for peer to peer protocols in general, attacks on the Distributed Hash Tables (DHT) based Kademlia routing scheme [1]or attacks on DHT itself [11] where the insertion of bogus data chunks into distributed data can also severely disrupt data distribution in BitTorrent-style networks. Most of these attacks are not very relevant for the distribution of free content. However, if BitTorrent is to be used for the distribution of commercial content attacks can result in financial loss, degradation of quality of service or violation of privacy. In this paper we concentrate on one requirement mainly relevant for the end-user and one for the ISP. The first is concerned with the quality of service, while the second focuses on the protection of data distributed vie the network. In addition, certain aspects of information governance are also to be addresses by protecting the content against eavesdropping and the privacy and secrecy of the end user related requests. Several other important requirements exist. Those include privacy requirements like the prevention of tracking of end-user behaviour, confidentiality of network parameters, accounting, etc. These requirements are not in the focus of this paper.

In today's systems for multimedia distribution these aspects are addressed by a tight coupling of the data to the node by encrypting the content especially for one particular node. The node stores one key known to the operator in the multimedia decoder hardware. This scheme does not allow for content redistribution.

A *trusted BitTorrent* requires a more sophisticated approach. First, all nodes that form the CDN have to be identified to ensure that data cannot be introduced into the network from unknown sources and that requests can always be associated with a registered user. However, identification is not sufficient. Nodes also need to establish trust into the correct behaviour of peers in order to prevent manipulated nodes to attack the network. It has to be prevented that malicious nodes inject false data into the DHT tables or centralised tracker. Furthermore, for the distribution to work peers have to make their own resources available. thus, free-riding on the peer to peer network by suppression of re-sharing needs to be prevented.

3 Trusted Computing essentials

Trusted Computing technology as defined by the Trusted Computing Group [13] is a technology implementing consistently behaving computer systems. This consistent behavior is enforced by providing methods for reliably checking a system's integrity and identifying anomalous and/or unwanted characteristics. These methods depict a trusted system's base of trust and thus are implemented in hardware, as it is less susceptible for attacks than software pendants.

To successfully realize stringently reliable modules, several cryptographic mechanisms are implemented on a hardware chip, namely Trusted Platform Module (TPM). This chip incorporates strong asymmetric key cryptography, cryptographic hash functions and a random number generator that is capable of producing true random numbers instead of pseudo random ones. Additionally each trusted system is equipped with a unique key pair whose private key is securely and irrevocably stored inside the chip. The chip itself is the only entity to read and use this key for e.g. signing or encryption.

This concept builds a basement for approving and establishing system integrity since it allows to truly trustworthy let a trusted system sign data and to securely encrypt data for one specific system. This is commonly used to measure system integrity and to ensure a system is and remains in a predictable and trustworthy state that produces only accurate results.

3.1 Trust for Measurement

The key concept of Trusted Computing is the establishment and extension of trust from an initially trusted security anchor up to other components of a system while boot-up. Each component loaded while booting up the system is measured before execution by computing a SHA-1 digest value of it. The first component of this cycle acts as security anchor and has to be initially trusted,

since it's integrity cannot be measured. This anchor is called *Core Root of Trust for Measurement* (CRTM) and is implemented as BIOS extension to be executed before any other BIOS code. Thus, the CRTM can measure the BIOS and the platform's firmware. Each subsequent component involved in the boot-up process thereupon measures its successive component. Each measurement is stored in form of hash-chains in *Platform Configuration Registers* (PCRs) on the TPM.

These hash chains in stored in PCRs allow to report the development of the system since the start of the CRTM. At system start, each PCR is initialized with zeros upon system start and then extended with measured data. Thus, other entities can analyze the current state of a remote system and the history since the last system start. This type of boot-up is called *Trusted Boot Process*.

3.2 Trust for Reporting

Another main concept of Trusted Computing is Remote Attestation, a process to prove trustworthiness of a Trusted Platform to an external party. To verify a platform's integrity, a subset of PCRs together with log of all measurements since startup (*Stored Measurement Log*, SML) is sent to the external party signed by the TPM with a so called Attestation Identity Key (AIK). The PCR values can then be compared with re-calculated values using the chronological order of measured components logged in the SML. Measurements include all events related to the start of software during the boot phase of a system and later on as part of the operation of the running system. From the SML no insight on the performed actions of loaded applications can be gained as it only documents that a certain software was started. AIKs represent pseudonymous identities. So-called privacy CAs certify that a particular AIK was generated in a TPM with a particular *Endorsement Key* (EK), the *Root of Trust for Reporting* (RTR) . The privacy CA also checks platform and EK certificates. The EK could identify a particular TPM and is therefore (for privacy reasons) not used for signing.

3.3 AIK Certification

Since each TPM is globally unique and thus identifiable and traceable, privacy issues arise when attesting a platform's state to external parties using Remote Attestation. In order to avoid this security issue, TPM chips provide for pseudonymity by allowing to generate temporary keys for attestation. These *Attestation Identity Keys* (AIK) can be created at any time using the `TPM_MakeIdentity` command and may be certified by a *Trusted Third Party* (TTP) to allow external parties to verify, that an AIK belongs to a TCG conform platform. AIKs can only be associated to their platform's EK by the TTP thus providing the platform with pseudonymity towards other entities. To issue an AIK credential, the platform has to send the EK-signed public key of a generated AIK key pair together with several credentials declaring the platform's TCG conformance to the TTP. After successful verification of the AIK and the platform's credentials, a particular data structure is sent to the platform. This

structure contains the AIK credential and can be securely loaded only into the TPM that signed the initial request using the `TPM_ActivateIdentity` command.

4 Notation

In the description of the protocol steps the following notation is applied. It is differentiated between symmetric, shared keys which are denoted as $K^{p,t}$ describing a shared key K between the peers p and t and asymmetric keys that are denoted as K^p_{pub} for the public portion of a key K of peer p and K^p_{priv} denoting the private part of key K from peer p.

Within the protocol definition special data structures for AIK certificates, denoted as $AIKCert^p$ for a AIK certificate of peer p and Stored Measurement Lists, denoted as SML^p for a SML of peer p, are used.

On top of the data structures the following operations are performed. Encryption of data using symmetric or asymmetric keys denoted as $enc\{data\}_{S^t_{pub}}$, stating that data $data$ is encrypted with key S^t_{pub}. Signatures are described $sig\{data\}_{S^t_{priv}}$ accordingly. The calculation of shared keys using the DH scheme is shown as $K^{p,t} = K^p_{priv} \circ K^t_{pub}$. The concept of quote signatures as they are introduced in Trusted Computing are denoted as $quote\{data, PCR_{0..n}\}_{AIK^p}$, showing that data and a set of PCR values are signed by an AIK of peer p.

5 Enhanced Protocol

To meet the security requirements stated in the security discussion above it is required to introduce strong identities, the afore introduced concept of remote attestation to satisfy the requirements of behavioural authenticity, and confidentiality of data transferred between the involved nodes. These security aims are to be established respecting the special needs of the peer to peer use cases like video on demand.

As presented before the BitTorrent protocol as the selected underlying protocol distinguishes the dispersion of the meta file to the nodes which is considered as an out of band operation, the tracker protocol, and the peer wire protocol. In the following, each of these three protocols is briefly revised and then extended using mechanisms from trusted computing.

5.1 Initial setup and meta file

Before the actual start of the protocol, each peer device has to have at least one valid AIK and corresponding certificates issued by a privacy CA. Keys and certificates can already be established during production or deployment. The AIK certificate states that a compliant TPM is installed on the respective device and that the privacy CA has checked EK and platform certificates. Furthermore, AIK certificates can also be used to provide additional information like that a particular TPM belongs to a specific network operator. Data to be accessed via

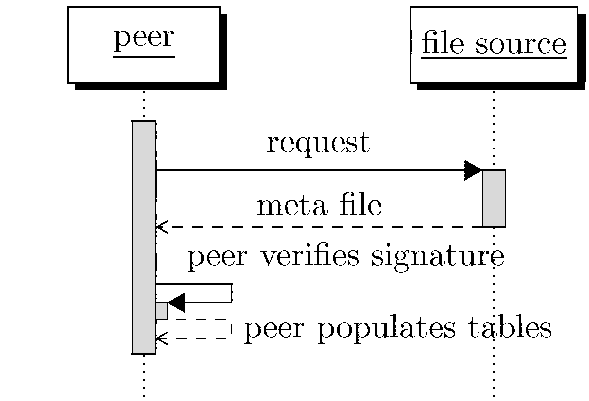

Fig. 1. meta file distribution

the network is associated with a meta file. This file provides information on the initial list of trackers to be connected.

In this context it is to be noted that the EK as the core identity of each TPM is assumed to be protected by the hardware itself and can not be changed or revoked. Therefore the identity of the device can not be changed during its lifetime. As the node is owned by the ISP use cases involving a change of the owner are not in the focus of the underlying trust requirements.

Further, the BitTorrent protocol distinguishes between centralised and distributed tracker by a differentiation in the meta file. In case of a distributed tracker for each tracker node credentials are included in form of public AIK and AIK certificate. Furthermore, each tracker needs to generate a key within it's TPM and certify it using the AIK. Such a certificate expresses that the private part of the key can only be used within this particular TPM. In case of a central tracker we assume that this tracker is in the control of the network operator. In order to explain different variants of the protocol we assume such nodes to be trustworthy. Therefore, a reduced protocol is used without attestation of the tracker. However, the identity of the tracker is still relevant and therefore, a key S^t_{pub} and its certificate is included in the meta file. This credential is used later on in the tracker protocol and is assumed to be created and protected by the owner of the tracker. To provide means to verify the integrity of the file it is signed by the issuer. By this signature each node is able to verify that the meta file is unchanged since its creation by the issuer of the content. Therefore no entity including the involved servers can not alter the meta file. The meta file distribution scheme is depicted in Figure 1. The server playing the role of the *file source* is assumed to be known to all peers.

5.2 Tracker protocol

Tracker collect information to locate nodes providing certain data. These requests include metrics from clients that help trackers to keep overall statistics about the torrent. The response of an tracker includes a peer list that helps a node to participate in the swarm.

As discussed in Section 2, establishing trust between nodes within this phase of the BitTorrent interaction requires authentication and a proof on the authenticity of the nodes behaviour, i.e. attestation of the state of the node. Furthermore, a symmetric session key is established between node and tracker. This symmetric key is rooted in the initial attestation session. Encrypted with this session key, the tracker provides a ticket to the node. This ticket is explained in detail in Section 5.3. Thus, mutual attestation (or attestation of the peer node in the case of a central tracker) establishes a trust relation between the overall network (represented by the tracker) and the node entering the network for the duration of the lifetime of the ticket.

As explained above we propose two different extensions to the BitTorrent protocol, one with mutual attestation and a reduced version with a trusted central tracker. Figure 2 shows the protocol for the centralised case whereas Figure 3 shows the distributed approach.

Both protocols have the same goal: a session shall be established that is rooted in the initial authentication and remote attestation between the nodes. At the end in each case a common symmetric key $K^{t,p}$ is established that is used to encrypt all following messages resp. their content. To create this common key the well know Diffie-Hellmann (DH) key exchange protocol is used [2] by transmitting the public DH keys K^{p}_{pub} and K^{t}_{pub} between tracker and peer. To prevent Man-In-The-Middle attacks both transmissions are either encrypted or signed.

For both variants the protection of the request in terms of privacy and secrecy has to be evaluated. Based on the result of this evaluation the request has to be encrypted or not. We assume here that the request is security sensitive and therefore show how to establish the session with protection of the request.

The centralised tracker approach as depicted in Figure 2 assumes that the tracker is operated under the direct control of the ISP or customer. Within the NaDa project, for example, all centralised services are operated in the protected perimeter of the respective operator. The identity of the tracker is bound to the keypair $S^{t}_{pub}, S^{t}_{priv}$ and optionally a matching certificate. S^{t}_{pub} and the certificate are provided in the signed meta file as shown above.

In step one peer p transmits an encrypted request to the tracker t using the key that is provided by the meta file. The requested content or service is denoted as *request*. Additionally K^{p}_{pub} as the public part of the DH key exchange protocol, the SML of p, and the AIK certificate of p are included to prepare key exchange and remote attestation. t then is able to compute a symmetric key $K^{p,t}$ (2). This can be done in parallel to the subsequent protocol execution.

t answers in step three with a data package consisting of K^{t}_{pub} for the DH key exchange. K^{t}_{pub} is signed together with K^{p}_{pub} using S^{t}_{priv} to vouch for the integrity

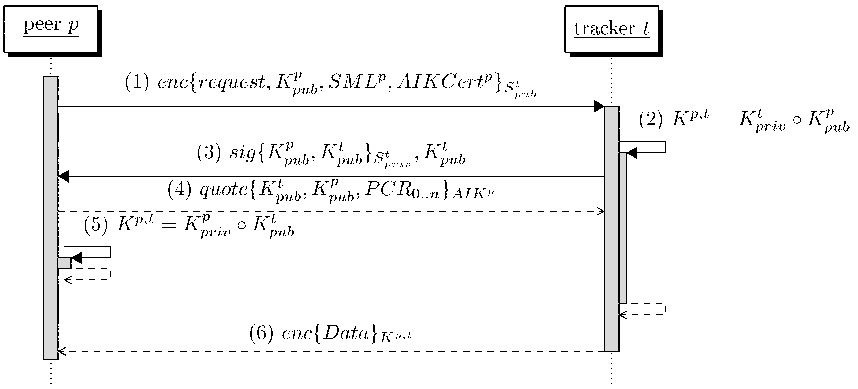

Fig. 2. Authentication and attestation of the peer p in the case of a tracker t operated by the ISP or content provider

and origin. By injecting K^p_{pub} we also grant for the freshness of the package in the view of the peer as the DH key is assumed to be freshly generated by the peer. It is to be noted that for simplification of key management the key pair S^t_{pub}, S^t_{priv} is used for encryption and signatures. However, different keys shall be used if a crypto algorithm is used where this is problematic.

Step four returns the TPM quote data using the K^t_{pub} to show the freshness of the remote attestation data. As part of the platform specification the selection of the appropriate PCR value is fix. The quote needs to ensure that the DH keying data K^p_{pub} was actually generated by the peer and not on any other device. Thus, we assume that this key is extended to a resettable PCR after the first step of the protocol. Note that usually only a small number of trackers is contacted at the same time and therefore this PCR extension does not induce efficiency problems. The tracker has now to decide based on the provided quote if the peer node can be assumed to be properly configured. In parallel, p can already compute $K^{p,t}$ (5). If the quote together with the produced SML results is accepted, t adds p to the list of active peers and returns (6) the result *Data* of the request encrypted by $K^{p,t}$. *Data* also includes the ticket to be used in the subsequent peer wire protocol. The transfer of the SML is part of the first step as also the SML should be transferred encrypted and additional encryption overhead is to be prevented.

The extension for the distributed protocol is shown in Figure 3. In this case mutual remote attestation allows for both communication partners to verify the

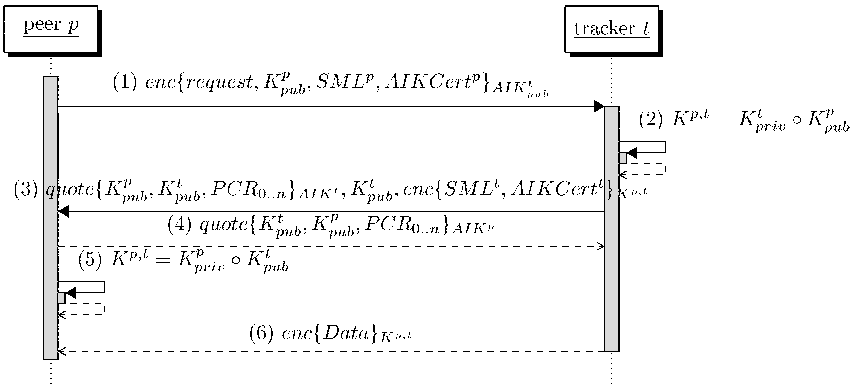

Fig. 3. Mutual authentication and attestation between tracker t and peer p in the case of distributed tracker

identity and authenticate the behaviour. To reduce the latency of the protocol we reuse the DH keys again for Nonces to prove freshness as we assume that each communication partner has control over the freshness of their DH keying information. For efficiency generation of the DH key may be deferred to idle periods of the respective node.

Step one is identical to step one of the centralised case except that a TPM-generated key is used for encryption. As AIKs cannot be used directly to encrypt data the meta file has to provide aside the AIK^t_{pub} also a second key and a certificate that states that this key generated by the TPM and cannot be migrated. Step three transmits the TPM quote of t, K^t_{pub}, and the by $K^{p,t}$ encrypted SML and AIK credential of t. p returns his quote in step four. Again, both parts of the DH keying information must be bound to the platform, e.g. by extending them to resettable PCRs. After the first four steps both communication partners verify the received quotes and SML lists. If the requirements of t are satisfied t transmits the requested result and ticket (6) encrypted by $K^{p,t}$. In steps (2) and (5) DH operations to calculate the shared key take place. It is to be noted that step (2) can be executed in parallel so that it does not consumes additional time.

The validation of the SML is non-trivial. It can use a data base for each suitable SML. More efficient is a scheme using a substitute in form of a credential issued by the ISP or a representative. Such a *system state ticket* states that a

certain PCR value is compliant to the desired state as defined by the operating party. Therefore, in this case it is not required to transfer the SML for each attestation.

5.3 Peer Wire protocol

The peer wire protocol (PWP) also requires the establishment of a trust relationship between the peers. Tickets established in the tracker protocol give evidence of the previous attestation by the tracker. Thus, trust establishment in the peer wire protocol can be reduced to secure authentication of both peers.

Each node participating at the peer to peer network has at least established one connection to a tracker where attestation was performed. This process is described in the previous section. Tracker only provide data on nodes that were attested.

The data transmitted in the tracker protocol is composed of the address of pB (e.g. IP and port), the AIK certificate of pB and the access ticket. This ticket is encrypted by the symmetric key negotiated between t and pB. It contains the AIK certificate of pA, the requested resource (e.g. the video requested), and the time of invalidation of the ticket. This scheme is similar to an existing proposal on TPM-based tickets [8, 9].

In step one of the enhanced PWP the *ticket* and K_{pub}^{pA} is transmitted to pB. Again, DH keying information shall provide freshness. pB answers (2) with his K_{pub}^{pB} and a signature on his K_{pub}^{pB}. For this we use the TPM quote command. Step three returns the signature on the K_{pub}^{pA} also using the quote command. Note that the choice of PCR registers is irrelevant as the platform shall only be authenticated. The DH keys need to be bound to the particular peers. As several instances of the PWP probably occur in parallel, using resettable PCRs is not possible. The ticket identifies a particular AIK to be used for the TPM_Quote. If this AIK is bound to the use within the PWP on the platform it cannot be used for quote commands invoked by other applications. This property of the application needs to be guaranteed in the state of the peer attested in the tracker protocol. After step three the signatures are verified and the symmetric key $K^{pA,pB}$ is computed on both sides (4). All further messages are then encrypted by this key (5). The enhanced PWP is shown in Figure 4.

PWP handshake is completed by sending a sequence of data to the contacted peer that consists out of pstrlen, which string length of pstr, as a single raw byte, (ii) pstr, the string identifier of the protocol, (iii) eight reserved bytes, (iv) info_hash, which is a 20-byte SHA1 hash of the info key in the metainfo file. This is the same info_hash that is transmitted in tracker requests. Finally (v) a 20-byte string used as a unique ID for the client. This ID is later be used by the contacted host to give the connection a unique identifier.

In the standard BitTorrent protocol the contacted client signals the acceptance of the connection by not closing the socket. It is not required that the contacted peer sends data to the requester during the handshake. To establish the trusted PWP additional data on the authenticity and a reply by the con-

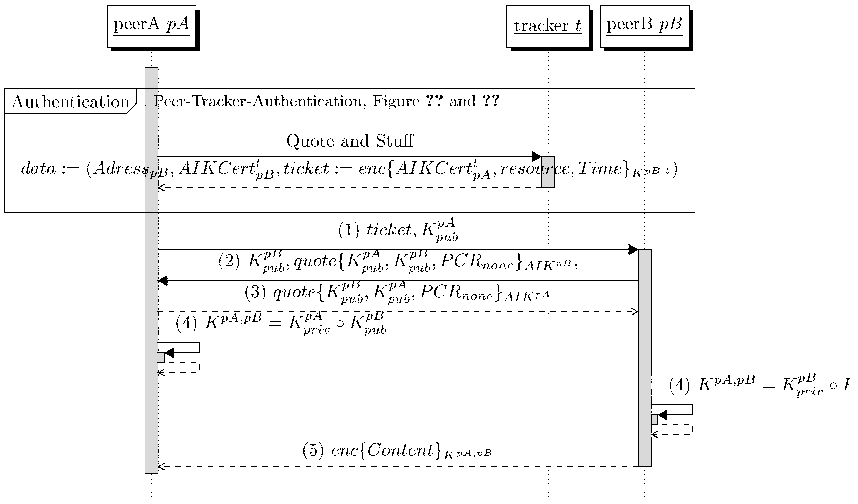

Fig. 4. Information flows between the entities of the trusted peer wire protocol

nected host is required. Therefore this different behaviour needs to be signaled to the other host. This can be done by setting a flag in the reserved eight bytes. The presented scheme is then added to this basic handshake.

For protection of the established symmetric key, the platform uses the TPM to *seal* the symmetric key to the trustworthy state of the node. In case of a reconnection between two nodes first the already established key is used. If the state of one node has changed e.g. due to a software update a new attestation is required and the symmetric key is replaced by a new one. For the case of a reconnection a special PWP variant is required.

Between two nodes the key exchange is only performed once as long as the system state is assumed not to be changed. Such a change may happen due to software updates. Therefore, for performance considerations both nodes store the key together with the AIK resp. ID of the node. Later contacts between these nodes are initiated first with an PWP handshake that is extended by Nonce, that the contacted node has to return encrypted. If this fails, the presented protocol is executed.

In case of distributed tracker it is to be considered that pA and pB are not necessarily connected to the same tracker. In this case different schemes are possible to solve the missing trust relation. Either t connects and attests each node that is known to t or t performs a search for the respective tracker where pB is connected to and receives an appropriate *ticket*.

6 Security Evaluation

The presented protocol provides a comprehensive solution for the challenge of establishing trust in the peer-to-peer protocol BitTorrent. The goal for this extension lies within the attestation of every peer participating in the file exchange, such that the QoS of the data delivery meets commercial requirements.

Within the standard BitTorrent protocol, there are already mechanisms included, that counteract an attacker's attempt to alter the data received at one peer from the others. This hash value of the file that resides in the torrent-description file has been unaltered for the trusted BitTorrent. However the possibility to disrupt or degrade the QoS of BitTorrent through malicious behaviour of one of the networks nodes is prevented.

The alliance of trustworthy peers is maintained by the tracker, that will require each new peer entering the network to not only authenticate itself reliably – through the strong association of a TPM's key with a certain physical platform – but also to attest its current configuration. The tracker may therefore firstly enforce strong access control to the network of peers based on the utilized peer's key, and secondly analyse the platform's configuration with respect to the trustworthiness of the client. The tracker may therefore prohibit malicious peers to enter the network, that would disrupt or degrade the QoS through malicious data flooding to the rest of the network or even the analysis of the current data distribution within the network, increasing the overall network load, but requesting huge amount of or very expensive file transfers from other peers.

The extension to the protocol between peer and tracker incorporates a standard Quote-Attestation of the TPM with an authenticated DH-KeyExchange. The major challenge here is the binding of the resulting DH-channel to the attested platform. In the case of trusted BitTorrent, this will be validated through an analysis of the SML at the trackers side. The certificates for the deployed software will guarantee for this. The second challenge is the enforcement of freshness within the DH run and subsequent protocol. For the trusted BitTorrent approach this is tackled through the use of fresh asymmetric key pairs on each run, where the public portion will be used as anti-replay nonce. Performance-wise the tracker can keep the same key parameter and therefore the creation of a new key can be performed within reasonable time.

The trusted peer wire protocol uses the same primitives of DH and TPM-Quote in order to establish trust among peers. Further it includes a Kerberos-like ticket through which the authorization of the requesting agent's platform is granted. The suit of Kerberos authorization tickets is well evaluated and the time information included in the ticket will prevent any replay after provision time has ended. The binding of the ticket to a specific peer is being performed through the TPM-binding of the data to a specific platform, such that the peers software also cannot be replayed to several other peers.

The propose protocol does provide a reasonable approach for increasing the QoS of a BitTorrent based file distribution. In case of high load at certain platforms, where more TPM-Attestation have to be performed than the TPM can provide, it is possible to include a more scalable solution [15].

Experience has shown that security protocols are error-prone and that a formal validation or verification of such protocols can reveal previously unknown attacks or increase assurance for secure protocols. Such a formal security validation would also be desirable for the trusted version of BitTorrent. However, existing approaches (e.g. from AVISPA) cannot be directly applied for several reasons. First, trusted BitTorrent aims at the establishment of trust relations between several parties and with respect to the status of the platforms involved. Such TPM-based properties cannot be directly described using the HLPSL of AVISPA. Furthermore, the protocol includes several relevant interfaces. Depending on the goals of the attacker, the interface between TPM and platform can be as relevant as the communication network between different platforms. Adequate attack scenarios have to be distinguished in order to validate the protocol in a realistic setting. Currently, more flexible specification and validation approaches based on the SH Verification Tool [6] are explored and extended on order to apply them to the protocols of trusted BitTorrent.

7 Conclusion

We presented an extension to the BitTorrent protocol that introduces strong identities and attestation of the state of communicating nodes. Through this extension to the protocol behaviour and in combination with trusted computing primitives malicious or manipulated nodes can be detected and suppressed from the peer to peer network. In the design of the extensions we considered aside the security requirements also possible optimisations w.r.t resource consumption.

Further research will be concerned with modifications of the presented protocols to give precise measurements on the consumed additional resources. Based on this analysis modifications in the protocols will be evaluated with respect to their performance. Furthermore, a security evaluation using the SH Verification Tool will be part of these steps to show which security requirements are fulfilled. This tool was already used to analyse parts of the TPM specification [6].

Based on the presented trusted BitTorrent we will evaluate possible application scenarios in the NaDa architecture. Hereby we focus on the establishment of trust relations and their use by virtualised environments running on a node.

References

1. I. Baumgart and S. Mies. S/Kademlia: A practicable approach towards secure key-based routing. In *Proceedings of the International Workshop on Peer-to-Peer Networked Virtual Environments*, 2007.
2. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on information Theory*, 22(6):644–654, 1976.
3. M. Einhorn and N. Madison. How Advertising and Peer-to-Peer are Transforming Digital Media and Copyright. *Journal of the Copyright Society*, 2007.
4. D. Erman, D. Ilie, and A. Popescu. Bittorrent session characteristics and models. *Traffic Engineering, Performance Evaluation Studies and Tools for Heterogeneous Networks*, 61(84):61.

5. W. Fang and L. Peterson. Inter-AS traffic patterns and their implications. *GLOBECOM-NEW YORK-*, 3:1859–1868, 1999.
6. S. Gurgens, C. Rudolph, D. Scheuermann, M. Atts, and R. Plaga. Security Evaluation of Scenarios Based on the TCG's TPM Specification. *Lecture Notes in Computer Science*, 4734:438, 2007.
7. B. Krishnamurthy, C. Wills, and Y. Zhang. On the use and performance of content distribution networks. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pages 169–182. ACM New York, NY, USA, 2001.
8. N. Kuntze, A. Leicher, and A. Schmidt. Implementation of a trusted ticket system. In *Proceedings of the IFIP Security 2009*, 2009.
9. N. Kuntze, D. Mähler, and A. U. Schmidt. Employing trusted computing for the forward pricing of pseudonyms in reputation systems. In *Axmedis 2006, Proceedings of the 2nd International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution, Volume for Workshops, Industrial, and Application Sessions*, 2006.
10. N. Laoutaris, P. Rodriguez, and L. Massoulie. ECHOS: edge capacity hosting overlays of nano data centers. 2008.
11. J. Liang, N. Naoumov, and K. Ross. The index poisoning attack in p2p file sharing systems. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12, 2006.
12. P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. *Proceedings of IPTPS02, Cambridge, USA*, 1:2–2, 2002.
13. C. Mitchell et al. Trusted Computing. *Trusted computing*, page 1, 2005.
14. J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. The bittorrent p2p file-sharing system: Measurements and analysis. *Lecture Notes in Computer Science*, 3640:205, 2005.
15. F. Stumpf, A. Fuchs, S. Katzenbeisser, and C. Eckert. Improving the scalability of platform attestation. In *Proceedings of the 3rd ACM workshop on Scalable trusted computing*, pages 1–10. ACM New York, NY, USA, 2008.
16. D. Wallach. A survey of peer-to-peer security issues. *Lecture Notes in Computer Science*, pages 42–57, 2003.