

Remote Attestation and Peer-to-Peer Networks

Ville Likitalo `liki@iki.fi`
Helsinki University of Technology
Laboratory of Information Processing Science

Abstract

This paper describes different remote attestation techniques, trusted computing and their suitability for a peer-to-peer network. A case study is made of a mobile peer-to-peer client on a Symbian platform.

KEYWORDS: Remote attestation, trusted computing, peer-to-peer, Symbian

1 Introduction

The growing use of peer-to-peer networks has raised a need to be able to block malicious clients out of the network. Clients should be able to identify malicious peers and isolate them from the network. Traditionally operating systems have been administrated by a reliable party, nowadays users most often have complete control of the operating system of the host and the host systems cannot be trusted.

Validating peers is particularly necessary for a peer-to-peer network. This is because if the network is unable to detect malicious requests and isolate malicious peers, an attacker would be able to effectively cripple the network easily, for example by sending invalid results. In a file-sharing network, an attacker could be able to abuse the network and perform further attacks, perhaps one similar to a “FTP Bounce”-attack. By tricking multiple hosts to reply to a request with false sender address an attacker could exploit the network to perform Denial of Service (DoS) -attacks. [3]

As remote attestation, we understand a way to remotely, or over a network connection, validate the operating system or an application running on the remote host. [10] Remote attestation has been proposed as a way to monitor the peers but if we cannot trust the operating system, the attestation protocol itself is vulnerable. Trusted computing is a proposal that tries to enforce trust on the operating system by ensuring that the operating system has not been tampered with. Trusted computing was proposed by the software industry and copyright owners as a way to fight piracy on the Internet and especially in the peer-to-peer networks such as Gnutella. This would be accomplished by ensuring that only software that respects digital rights management systems could be executed or access the protected material, but ironically enough the very same technique is required to secure the peer-to-peer networks against malicious clients.

Traditionally, attestation has been performed by comparing the image of the program against a cryptographic hash of a valid image. A chain of cryptographic certificates ensures that trusted hardware has loaded a valid boot image and then

a valid operating system. The certified operating system then produces a certificate for the software in question. This is however not quite feasible when considering an unmanaged and de-centralized network of equal peers. [10]

In this paper we shall assume a de-centralized network of equal peers when talking about a peer-to-peer (P2P) network. The P2P network can be either a centrally managed one, such as Napster, or an unmanaged one and purpose of the network may be anything ranging from exchanging data between the peers to searching for extra-terrestrial life. We shall only assume that the result given by a peer as a response to a request can be verified by an other peer as well, given the same request.

We shall assume that each client on the network runs on a separate host and that the threat originates from one or multiple malicious hosts. We shall also assume that the attacker is able to control the network that is used to transfer data between the peers, thus we can use the traditional Dolev-Yao threat model, [4] extending it with the notion that the attacker is also able to control the operating system of the malicious hosts.

Our goal is to identify whether remote attestation techniques can be used to secure a P2P network and how the different techniques can be exploited the best, not forgetting that a combination of different techniques could well yield the best results. The requirements are that the network must be able to isolate malicious clients and accept different versions of the software on different host platforms as valid. Also, the attestation protocol must not be computationally feasible to break and multiple malicious clients must not be able to evade isolation or cause one or multiple legal peers to become isolated.

First, we shall begin the study with an unmanaged P2P network in mind and finally make a case study of a mobile P2P client running on a Symbian platform. In that case we will assume that the operator will act as a trusted 3rd party and centrally manage the network if required, assuming that different operators will interact transparently or that only one operator exists.

2 Trusted computing

When pirated digital copies of music and movies began to spread in the Internet like wildfire, copyright holders such as Disney eventually woke up to the problem, and Digital Rights Management (DRM) was quickly innovated as the solution. The original idea was that your personal computer would refuse to play illegal copies and only allow operations approved by the copyright holders with legal copies.

However, only a slight problem existed - There was absolutely no way to stop users from running programs that would not care about these restrictions. The Trusted Computing Group¹ was formed and they took the idea presented by Bill Arbaugh, Dave Farber and Jonathan Smith [2] into exploitation. Namely, if the computer would only boot an operating system that is known to respect DRM restrictions and it could ensure that the operating system image has not been modified, the problem would be solved.

In this context, trusted hardware works in a manner that the computer includes a secure chip storing encryption keys that are used to encrypt DRM protected material. If the bootstrap loader determines that the operating system has been booted into an acceptable state and all hardware is DRM capable, it will release access to the keys. The keys are then used to decrypt the protected material. [1]

Further, the trusted platform could be capitalized on enforcing software licenses. If the bootstrap loader would only load a valid operating system, it can also ensure at the same time that the license for the operating system is valid. And the systems offered by the DRM capable operating system can then be used to ensure that other application licenses are also valid. [1]

The trusted platform is worthwhile not only to the power-hungry copyright owners and software vendors but the very same methods used by the DRM systems can be used to secure a much stronger access control to classified material. Which is certainly something that most large companies and organizations would love to have with commodity hardware, instead of having to pay for expensive secure systems and installations. [1]

Currently the Trusted Computing (TC) projects of interest are Palladium by Microsoft, SELinux by NSA and TrustedBSD. The last two projects are not totally related with the DRM-concepts but we will not go into details in this paper.

2.1 Remote attestation

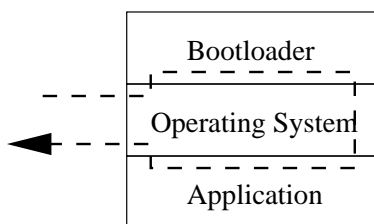


Figure 1: Remote Attestation

Remote Attestation is a way for an application to authenticate itself to a remote party, or for a remote party to verify the authenticity of the application. [10, 8]

And how is the Trusted Computing related with Remote Attestation (RA)? As the name says, RA is performed over a network connection and as network connections are handled by the OS, we cannot trust the attestation protocol if we cannot trust the operating system. This is illustrated in Figure 1. TC enforces trust on the OS, and then we can rely on the attestation protocol to function as required. [10]

Next we shall discuss different methods suitable to perform RA and their usability in a P2P network context.

2.1.1 Cryptographic attestation

The traditional way to perform remote attestation has been to use cryptography. The validating party would compute a cryptographic hash² of the image of the application to be validated. The validating party, namely the OS, would then sign this hash and the requester would compare the received hash against a hash of a known valid image. Although the hash would only validate contents of the image of the program and not what the program does computationally, it would serve its purpose in this context unless hash collisions are relatively easy to come up with. [10]

The protocol could be enhanced with bit commitment to ensure that the computation of the hash was actually performed. However, it would still not guarantee that the hash was computed of the memory print of the running program. Thus it is evident that we must have trust on the OS. [10]

In a P2P context the hashes are also problematic. The hash result is dependent on the image, different platforms and versions of the software would all result in a different hash. The number of valid hashes would then be the number of target platforms multiplied by the number of software versions as shown in equation (1) where R is the number of different software revisions, or versions, and P the number of target platforms. While such a list would certainly not be impossible to upkeep, it would be sensible to look for alternative solutions, especially if there are many target platforms or the client is an open-source application as the number of different valid revisions could grow rapidly. [8]

$$R * P \quad (1)$$

2.1.2 Semantic attestation

Semantic attestation is a concept proposed by Vivek Haldar, Deepak Chandra and Michael Franz [8] aiming to overcome the shortcomings of the traditional cryptography-based attestation model. Haldar, Chandra and Franz propose a way to use a virtual machine to enable attestation based on high-level application properties in a platform-independent way, which makes it much more suitable to be used in a P2P context. [8]

Cryptographic attestation methods can be used to verify that some particular version of an application is running on the host, but we would be more interested in whether the application in question abides by some security policy. By using a Trusted Virtual Machine and a platform independent byte code that includes also high-level information about the application,³ Haldar and others were able to define a system that allows meaningful attestation of applications. Semantic attestation allows: [8]

- Attesting properties of classes.
- Attesting dynamic properties.
- Attesting arbitrary properties.

²Such as MD5 or SHA1

³Such as Java byte code

¹Microsoft, Intel, IBM, HP and AMD

- Attesting system properties.

This allows semantic attestation to overcome the most critical problems of the cryptographic attestation. Semantic attestation can be used to ensure that the application and its environment fulfills the conditions necessary for supposed behavior and as the system is platform independent and the verification is done based on the properties of the application, different versions of the application cause no additional problems. Naturally, provided that the application has not changed too much. [8]

However, the requirement of the trusted platform still exists for the semantic attestation as well. Which is of course totally natural as successful attestation is still dependent on the OS, or this time the virtual machine. [8]

Considering a P2P network again, semantic attestation is a huge improvement over cryptographic attestation. as Haldar and others present with an example of a Gnutella-network enhanced with semantic attestation. By using semantic attestation, the P2P network is able to also prevent impersonation and peers from providing false content, thus preventing two major vulnerabilities presented by Steven M. Bellovin. [3]. When using only cryptographic attestation, an attacker could still be potentially able to cause the client to impersonate as an other client and nothing could stop the attacker from providing false content. [8]

2.1.3 Reputation models

Reputation based attestation is a fundamentally different model from the previous two, described by Michael Schillo, Petra Funk and Michael Rovatsos. [12] Reputation based attestation works by having reputation of entities upkept in a repository and if the reputation of an entity declines under a certain preset boundary, that particular entity becomes isolated from the others. [12]

In a P2P network this concept is somewhat manageable as long as the network is managed by a central trusted entity, such as Napster, or there exists a trusted 3rd party that acts as a trust repository. If the P2P network is unmanaged, then the trust repository would have to be a distributed hash table, or a similar decentralized data structure.

Assuming that we would store tickets – negative feedback – about malfunctioning clients, for example a client providing false content, into the trust repository. Then, if we would require two tickets by different parties would to isolate a malfunctioning client we would be able to prohibit an attacker using a single host from exploiting the system. With multiple malicious peers an attacker could however cause legal peers to be isolated from the network. While requiring more tickets would mean that the attacker would need more malicious peers, at the same time the barrier to isolate a malicious peer would grow higher. And if the system does take not only negative feedback into account but also positive one, it would be easier for the attacker to evade isolation by submitting false positive feedback when required.

What is left is a formal proof that with this model we are never able to totally factor out the impact of an attacker exploiting multiple malicious peers. This is because it is impossible to select the “witnesses”⁴ and the trust repository

⁴Ticket granters

maintainers from amongst the legal peers only, which is covered more formally by Bin Yu and Munindar P. Singh. [14]

Despite its shortcomings, reputation based attestation is of practical interest as it can be implemented totally on the application level and no trusted platform is required.

2.1.4 Proof-Carrying Code

Proof-Carrying Code (PCC) is a concept presented by George Necula. [9] Producers are required to provide a formal safety proof for the code in question, so that clients are able to analyze and validate the code before executing it. With remote attestation, we try to determine whether a client is executing valid code, but with PCC a client can verify that it has received a valid executable that can be safely executed. While not directly related to remote attestation, the ability to verify that an unknown executable is safe to execute might be of interest in our study. [9]

Although PCC requires someone to supply the formal proof for the code and generating the proof can be tedious, validating the code itself is relatively simple for the client. However, we must note that PCC does not actually prohibit anyone from intentionally running malicious code. [9]

3 Remote attestation in a peer-to-peer network

And how would a P2P network benefit from remote attestation technologies then? A P2P network that cannot validate its peers and isolate malicious ones is totally at the mercy of good-willingness of its peers. As presented by Steven Bellovin [3] an attacker can exploit the network for further attacks, or effectively cripple the usefulness of the network by polluting it with false content.

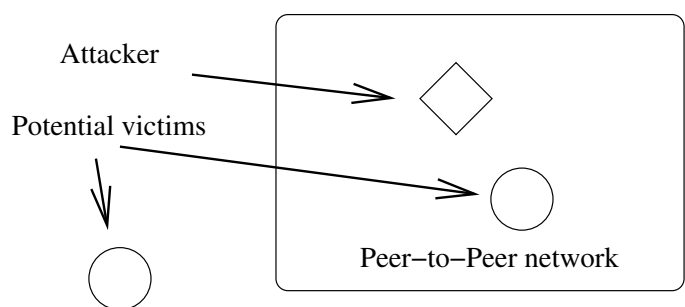


Figure 2: Attack layout

In a typical attack scenario the attacker is itself a part of P2P network as illustrated in Figure 2 and the attacker would then exploit the good-willing parties of the network. Known examples of these kind of attacks involve for example tricking many hosts to send a large response to a wrong receiver - Who becomes a victim of the denial-of-service attack - by using specially crafted requests. [3]

Reputation based attestation models will not solve the issue for decentralized unmanaged P2P networks, but they can be successfully used if a trusted 3rd party exists. In that case the trusted 3rd party can witness the tickets and an attacker cannot cause isolation of a legal peer. However, a malicious

peer may be able to evade isolation if it is able to detect requests by the trusted 3rd party and delude the witness.

Semantic attestation as proposed by Haldar, Chandra and Franz is a very suitable method to be used in a decentralized P2P network, such as Gnutella, as presented by the authors. Although trusted computing is not yet readily available, the pressure built up by the copyright owners and software vendors is pushing it forwards. Therefore we should not be surprised to see it in commodity hardware, despite the constant and active resistance by various consumer organizations. Motivation for the trusted computing is to fight piracy on the Internet but the very same technique is required to secure the file-sharing P2P networks, that are widely used to trade pirated copies of music and movies, against exploitation or becoming crippled.

As an alternative to semantic attestation, a combination of cryptographic and reputation based attestation is also possible. Although cryptographic attestation requires dealing with the different hashes, it can be extended by a reputation based model to provide protection against malicious peers supplying false content, as cryptographic attestation itself does not prevent a malicious peer from potentially impersonating as an other peer or polluting the network with false content. Cryptographic attestation will protect the reputation based model against false tickets from malicious clients, because the tampered application code is positively identified, and reputation based model will protect the network against peers poisoning the network with false content. Cryptographic attestation will also protect the trust repository from being stored on tampered peers.

3.1 Mobile peer-to-peer client

Mobile P2P clients are of particular interest as computational resources are scarce on mobile platforms. A mobile client with for example the Symbian operating system is not far from a trusted platform, although the Symbian OS still lacks for example in memory protection. To be able to use cryptographic or semantic remote attestation, the mobile clients would have to be able to connect into each other, which is not a desired requirement in the mobile world, often it is even impossible.

What is important to notice, is that there always exists a trusted 3rd party to act as the trust repository and witness required to make the reputation based attestation model functional. Mobile P2P clients would in any case be more or less reliant on the operator and as most services would also be provided by the operator, we can assume that operator will also act as the trusted 3rd party.

It is also important to realize that authentication and identification services are readily available at the mobile world. Although these will not stop potential attackers but the real culprits behind the attack can be traced and brought to account. As is not the case in the Internet where it is often impossible to identify the culprits. At least this factor should have an effect on the number of attacks.

Remote attestation techniques are of spectacular interest in the mobile world. As the mobile clients develop further, it is becoming more important for the operators to validate the operating system of the phone, and perhaps the applica-

tion clients as well for services provided by the operator as well. In a P2P network cryptographic attestation would be unfeasible, but it would suit the operator's needs well.

Although validity of a hash acquired by cryptographic attestation could be verified from the operator, eliminating the need to store all the different valid hashes in the client, reputation based attestation would seem much more suitable to be used in the mobile world.

3.1.1 Proof-Carrying Code and Reputation Model

Joan Feigenbaum and Peter Lee present [5] how a reputation based model and proof-carrying code can be combined to distribute mobile code securely. Author of the code requiring distributed computational resources would supply the formal proof for the code in question and others could decide whether to allow the code to be executed based on the verification and the trust repository. For example, this model could be used to securely trade applications between mobile clients, provided that the proof-carrying code mechanisms are present. Further it could be used in a P2P network in which clients would share computational resources when requested, as Feigenbaum and Lee present. [5]

4 Conclusion

Considering remote attestation in a peer-to-peer network context, we conclude that semantic attestation offers several advantages over the other models, provided that a trusted platform is available. However, a combination of cryptographic attestation and proof-carrying would be a very suitable solution for a mobile operator. A reputation based model does not require a trusted platform if a trusted 3rd party is available and therefore is the only viable alternative for current solutions so far.

5 Acknowledgments

I would like to thank Mr. Dan Forsberg for tutoring this article.

References

- [1] Ron Anderson. Trusted computing frequently asked questions. 2003.
- [2] Bill Arbaugh, Dave Farber, and Jonathan Smith. A secure and reliable bootstrap architecture. In *IEEE Symposium on Security and Privacy*, pages 65–71, 1997.
- [3] Steven M. Bellovin. Security aspects of napster and gnutella. *USENIX*, 2001.
- [4] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, March 1983.
- [5] Joan Feigenbaum and Peter Lee. Trust management and proof-carrying code in secure mobile-code applications. *DARPA Workshop on Foundations for Secure Mobile Code*, 1997.

- [6] Michael Franz, Wolfram Amme, Matthew Beers, Niall Dalton, Peter H. Fröhlich, Vivek Haldar, Andreas Hartmann, Peter S. Housel, Fermín Reig, Jeffrey von Ronne, Christian H. Stork, and Sergly Zhenochin. Making mobile code both safe and efficient. *OASIS: Foundations of Intrusion Tolerant Systems*, pages 337–358.
- [7] Michael Franz, Deepak Chandra, Andreas Gal, Vivek Haldar, Ferming Reig, and Ning Wang. A portable virtual machine target for proof-carrying code. In *ACM SIGPLAN 2003 Workshop on Interpreters, Virtual Machines and Emulators (IVME03)*, 2003.
- [8] Vivek Haldar, Deepak Chandra, and Michael Franz. Semantic remote attestation - a virtual machine directed approach to trusted computing. In *3rd Virtual Machine Research and Technology Symposium (VM '04)*, pages 29–41. The USENIX Association, 2004.
- [9] George C. Necula. Proof-carrying code. In *Conference Record of POPL '97: The 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 106–119, Paris, France, jan 1997.
- [10] Siani Pearson and et al. *Trusted Computing Platforms: TCPA Technology in Context*. Prentice Hall PTR, 2002.
- [11] Geetha Ramachandri and Delbert Hart. A p2p intrusion detection system based on mobile agents. In *the 42nd annual Southeast Regional Conference*, pages 185–190. ACM Press, 2004.
- [12] Michael Schillo, Petra Funk, and Michael Rovatsos. Using trust for detecting deceitful agents in artificial societies. *Applied Artificial Intelligence*, 14(8):825–848, 2000.
- [13] Arvind Seshadri, Adrian Perrig, Leendert van Doorn, and Pradeep Khosla. Swatt: Software-based attestation for embedded devices. In *IEEE Security and Privacy Conference*, 2004.
- [14] Bin Yu and Munindar P. Singh. An evidential model of distributed reputation management. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 294–301. ACM Press, 2002.