Trust enforcing and trust building, different technologies and visions

Michele Tomaiuolo
Department of Information Engineering, University of Parma, IT

ABSTRACT
Concern about vulnerabilities of IT systems is growing together with attention to risks of intrusive cyber-control over personal activities and data. This article discusses some new technologies that are being integrated into computing devices for realizing so-called Trusted Computing and Digital Rights Management systems, which can remotely attest their current hardware/software state and can enforce external policies to access protected content. These technologies are then confronted with distributed Trust Management systems, which realize access control for local resources on the basis of delegation of access rights according to local trust decisions. Both technologies are discussed from various points of view: architecture, vision, ethics, politics and law.

Keywords: security; trust; delegation; authorization; Web services; intellectual property rights; information access; digital books; multimedia

INTRODUCTION

The growth of attacks to computing systems, as well as the privacy issues emerging in many popular communication technologies, are attracting more attention to the overall security of ICT systems and infrastructures. A central issue is the protection of resources from unauthorized access, which relates to the additional issues of secure user authentication, definition and enforcement of access policy, reliable accounting, confidentiality and integrity of data.

A new wave of technologies and standards, in particular, deals with the enforcement of a “trusted environment” on a wide range of devices, from embedded and pervasive systems to servers. Such technologies are often referred to as Trusted Computing (TC) and have the declared objective of executing applications and storing data in a secure way. Intel, for example, cites the rising tide of malware and other attacks through malicious programs as one of the main motivation for its new Trusted Execution Technology (Greene, 2012). However, the environment set up by Trusted Computing technologies is also fit for implementing Digital Rights Management (DRM) systems. In fact, a “trusted system” can attest to copyright holders that access policies for protected media files will be earnestly enforced.

Another approach to access control is based on decentralized Trust Management (TM), i.e. on the peer-to-peer delegation of access rights among users. In Trust Management systems, in fact, the administrator of local resources is considered as the ultimate source of trust about them, and is provided with means to carefully regulate the flow of delegated permissions. No a-priori trusted parties are supposed to exist in the system, in general. This can also represent the basic setting for improved interoperability among diverse systems.

This article presents both TC and TM architectures, which in some sense are orthogonal and also complementary. However, their visions are at contrast, with regards to overall centralized or hierarchical control and user empowerment. The following section provides an overview of DRM technologies, Trusted Computing architectures and the supporting legal framework for dealing with infringements. Then, the principles of Trust Management for peer-to-peer delegation and some strategies and issues of Trust Negotiation are described. Also, the dDelega library is introduced, as an example of use of the distributed delegation and negotiation mechanisms of TM in the context of Web services. Finally, a critical comparison between the different approaches to trust building and enforcing is provided, together with other concluding notes.

TRUSTED SYSTEMS

Regarding the basic architecture and functioning of Digital Rights Management systems, various so-called "Rights Expression Languages" have been proposed, for the management of digital rights for media content distribution. These languages and frameworks are essentially the result of efforts of businesses to protect digital material from reproduction and sharing. However all Rights Expression Languages just allow copyright holders to express restrictions about the usage of a resource (for this reason, critics of those technologies often refer to them as "restrictions expression languages"), without being able to enforce by themselves the policies they convey. The usage of "trustworthy" systems (Coyle, 2003) and the application of international laws is necessary for actually enforcing the policies these languages allow to express.
In fact, obfuscation is the Achille's heel of most DRM systems (Stamp, 2003). Obfuscation is necessary for the realization of DRM restrictions on common PCs and other open systems, to make reverse engineering more difficult and protect in some way the decryption function. But in traditional cryptography, obfuscation has always been considered a poor solution, with uncertain resistance to attacks. Moreover, in open systems the decryption function (generally a cryptographic key) can be gathered by scanning the system memory at runtime.
To overcome this problem, content producers are encouraging laws against circumvention of DRM policies. But another parallel effort is directed toward the realization of so-called Trusted Computing systems, composed only of approved hardware and software components, which can assure the respect of media access restrictions.

Digital Rights Management

DRM infrastructure is being included into a growing number of devices and systems, with the support of ICT firms, together with content producers and distributors. In computer systems and other devices, DRM technologies can be found at different levels: (*i*) hardware, e.g., the Content Scramble System (CSS) of DVD players; (*ii*) operating system, e.g. the Rights Management System (RMS) of Microsoft Windows Server 2003; and (*iii*) application, e.g. Apple iTunes. However, current DRM systems are mostly not interoperable and in many cases they either do not satisfy the expectations of customers or do not prevent unauthorized accesses. At this regard, one of the latest news is the specification by the W3C of a set of standard APIs, named Encrypted Media Extensions (EME), allowing HTML5 applications to interact with different protection modules, which will implement the core DRM technology (Halpin, 2013; Bright, 2013).
Among Rights Expression Languages, XrML and ODRL are worth mentioning, as being both proposed as generic frameworks for the management of digital rights for media content distribution (XrML, 2002; Iannella, 2002). Both XrML and ODRL are XML-based languages, built on previous works made at Xerox PARK, which resulted in the definition of the Digital Property Rights Language (DPRL, 1998). In fact, the real difference lies in their intended domain: while XrML has a broader applicability, instead ODRL seems more oriented to the specific media publishing market. In fact it specifies media formats, resolutions and frame rates, while XrML doesn't.
One of the most important cases of DRM inclusion into widespread technologies is MPEG, the standard from the Moving Picture Experts Group for multimedia applications. The MPEG-21 version of the standard defines three technologies to deal with DRM: Intellectual Property Management and Protection (IPMP), Rights Expression Language, Rights Data Dictionary. Specifically, MPEG-21 defines a standard Rights Expression Language for communicating access policies from content creators to content consumers. The language is directly based on XrML, to which it adds a precise dictionary of terms and permissions for describing and accessing multimedia content. In MPEG-21, a

digital item is defined as the fundamental unit of distribution and transaction, which users can interact with. Thus, MPEG-21 itself can be defined as a standardized technology for allowing users to access, exchange and manipulate digital items.
Another application area of DRM is the eBook market. In general, protected eBooks have generated concerns due to unclear policies, on the one hand, and piracy issues, on the other hand. The market is still fragmented, with different competing formats. Adobe PDF remains one of the preferred formats for viewing and printing electronic documents. The reader software, available on a wide range of platforms, includes and implementation of DRM. However, the PDF DRM management has been proven quite weak, above all for its obfuscation method.

Trusted Computing

According to the definition provided by the Trusted Computing Group (TCG, 2007): "Trust is the expectation that a device will behave in a particular manner for a specific purpose." It is worth noting that the reference to "expectation" is kept in the vague, possibly intending either the local user's expectation, or that of some remote entities (Arbaugh, 2003), including service providers in a SOA scenario, or copyright holders in a DRM scenario.
In its more common acceptation, the concept of Trusted Computing can be traced back to Balacheff et al. (2000) and Chen et al. (2000). The first specifications about a Trusted Computing system were released in 2001 by the Trusted Computing Platform Alliance (TCPA) (Pearson, 2002; Pearson & Balacheff, 2003). Currently, the Trusted Computing Group (TCG) has inherited the work and aim of the original alliance and is continuing the development of those specifications. Quite expectedly, the specifications generated many concerns about privacy and control (Reid et al., 2003). More in general, the principles at the basis of Trusted Computing have attracted many critical comments (Anderson, 2003; Arbaugh, 2003; Walker, 2003; Stallman, 1997; Stallman, 2002; Schneier, 2002; O'Riordan, 2006), some of which have became quite popular by themselves.
Among the main features that are provided by a Trusted Computing system, there are:

- Verified launch. A hardware-based process of system startup can verify the state of the Measured Launched Environment (MLE). The verification creates a chain of trust through consecutive cryptographic measurements, involving hashing and signing algorithms. A policy engine can confront the system measurements with a Launch Control Policy (LCP), for determining if the state is acceptable.
- Protected storage. Some methods are specified to protect both confidentiality and integrity of data on mass storage, through cryptographic keys protected by the Trusted Platform Module (TPM). Those methods assure that the platform configuration is adequate to make the data accessible, e.g. it is running the same application, possibly with some additional software.
- Platform attestation. The system can issue credentials attesting platform measurements. Attestations can be required by the local user or by remote entities to verify the compliance with existing policies and to audit activities. Attestation may be used to assure that a certain media file will not be used beyond the limitations imposed through some form of DRM by its system of origin, or that the system will correctly conform to an agreed protocol.
- Virtual Machine isolation. The system is protected from unauthorized direct memory accesses (DMAs). Application and data isolation is enforced on the system, to support the safe execution of software. Data stored into the "curtained memory" can only be read and modified by its owner process, so that data and code can be protected from any interference by other software and in particular from reverse engineering attempts.

These features require some support from the underlying hardware, which includes new or strengthened features to be provided by processors, chipsets and additional modules.

- CPU. It provides support for simultaneous protected partitions, which provide the isolated execution of some processes through hardened access to memory and other system resources. The architecture allows the execution of different parts of an application in different partitions, either standard or protected.
- Trusted Platform Module (TPM). It is responsible for storage of the root cryptographic key pair and other security data. It supports the attestation process, to confirm the state of the system is acceptable, according to a certain policy.
- Chipset. It interfaces to the TPM and enforces memory and data protection policies. It provides protected channels to graphics hardware and I/O devices, securing data transfers on behalf of the protected partitions.
- Keyboard and mouse. Input data is encrypted using a secret key, shared between the input device and the input manager for a protected partition.
- Graphics. The pathway between an application and the output display context (e.g. a screen window) is protected, to avoid the information being observed by other unauthorized processes.

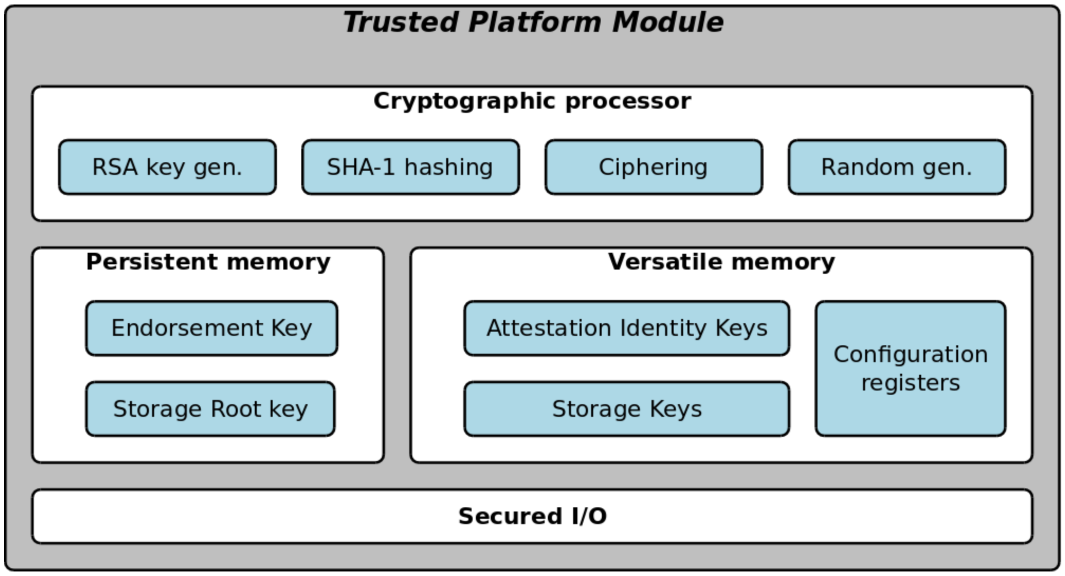

Figure 1. Internal components of a Trusted Platform Module

TCG also specifies a trusted boot process, on the basis of previous analysis and results (Tygar and Yee, 1991; Clark, 1994; Arbaugh, Farber and Smith, 1997; Itoi et al., 2001). The state of the system at startup time can be measured and attested by the TPM. A secure boot process of this kind complements the specification of UEFI (Unified Extensible Firmware Interface), the standard firmware interface for modern personal computer. UEFI Secure Boot assures that only “secure” software is loaded, i.e. code with a known hash or signed with a cryptographic key acknowledged by the system.
One of the largest projects in the field of trusted systems is the Next-Generation Secure Computing Base (NGSCB), the security architecture proposed from Microsoft for Trusted Computing. It was initially published under the name Palladium, and is planned for inclusion in common PCs and other devices (England et al., 2003; England & Peinado, 2002; Peinado et al., 2004). NGSCB is described as a secure operating system, requiring a precise set of features from the underlying hardware platform. The software components of NGSCB are of two types: (*i*) the Nexus kernel, a security component that is part of the Operating System; and (*ii*) Nexus Computing Agents, trusted modules within NGSCB-enabled applications. The hardware parts of NGSCB are essentially those specified by the

TCG.
Apart from NGSCB and TCG specifications, other similar systems are being developed, such as: Terra (Garfinkel, Rosenblum & Boneh, 2003), Perseus (Pfitzmann et al., 2001; Sadeghi & Stüble, 2004), the Open Trusted Computing architecture (Kuhlmann et al., 2006), and the European Multilaterally Secure Computing Base (EMSCB) (Sadeghi, Stüble & Pohlmann, 2004). At the core of all these systems, there are components and protocols which resemble the recent versions of NGSCB.

Legal context

The diffusion of digital contents and DRM systems is accompanied by a certain development in the legal framework, too. Since the market of digital content has a global reach, also the legal updates are being discussed in international forums. The main international body in this field is the World Intellectual Property Organization (WIPO), which was created in 1967 and entered into force in 1970. Its aim is "to encourage creative activity, to promote the protection of intellectual property throughout the world" (WIPO, 1967). In 1974 it became a specialized agency of the United Nations and its headquarter is now in Geneva, Switzerland. WIPO currently has 184 member states and administers 24 international treaties. Most of UN members have already joined WIPO. The Bureaux Internationaux Réunis pour la Protection de la Propriété Intellectuelle (BIRPI) – which had been established in 1893 to administer the Berne Convention for the Protection of Literary and Artistic Works and the Paris Convention for the Protection of Industrial Property – can be considered a predecessor of WIPO.
In 1996 the WIPO Copyright Treaty (WCT) was approved. It is one of the main results of the international body and is being implemented by many of its members. It requires WIPO members to pass bills against the circumvention of DRM mechanisms. Among the main enactment of the treaty, there are the Digital Millennium Copyright Act (DMCA), passed in the USA in 1998, and the European directive on copyright (EUCD), passed in the EU in 2001.
The DMCA is the act which criminalizes the circumvention of DRM and other measures implemented to protect copyrighted material. The law is infringed by the very circumvention of DRM mechanisms, whether or not copyrighted material is actually accessed. The DMCA also increases the penalties for unauthorized access to copyrighted material through communication networks and the Internet in particular. It extends the reach of copyright, but it reduces the liability of service providers for copyright infringement by their users. After being passed unanimously by the U.S. Senate, the act was signed into law by President Bill Clinton on October 28, 1998.
The EUCD is a directive of the European Union which addresses the same issues as the DMCA. A separate directive, the Electronic Commerce Directive, addressed the exemption from liability (both direct and indirect) of Internet service providers. The Copyright Directive is also known as the Information Society Directive and officially is identified as the Directive 2001/29/EC of the European Parliament. It was enacted under the internal market provisions of the Treaty of Rome. Its execution requires a separate legislation within each member state of the Union. Due to some vagueness of the EUCD directive, which does not deal with all cases, and public attention to copyright laws, the transposition has not been entirely linear, with six member states being taken to Court of Justice for failure to implement the directive in time.
More recent copyright proposals include Stop Online Piracy Act (SOPA), PROTECT IP Act (PIPA), Anti-Counterfeiting Trade Agreement (ACTA), and Trans-Pacific Partnership Agreement (TPP), which are also generating many worries for their wide span and vague terms and are the argument of flamed debates (Carrier, 2013). The protests against SOPA and PIPA in 2012 involved the temporary closure of major websites, including Wikipedia.

Privacy and security concerns

According to various authors, the diffusion of trusted computing systems could lead to worrying scenarios (Walker, 2003; Anderson, 2003; Stallman, 1997; Stallman, 2002; Schneier, 2002; O'Riordan,

2006). For example, the possibility to lock out a user from accessing certain files or software products, on the basis of remote choices, can be used by companies to acquire a competitive advantage. In fact, if files were to be encrypted by an application itself, then using that particular application would become a requirement, to access those files. The effect would be independent from the ability of other applications to handle a particular file format. But certain authors expressed even tougher preoccupations. In fact, the local enforcement of remote policies, once made possible, could not just be used for DRM control, but also pave the way for social control and limitation of users' abilities. This possibility could be directed to avoid criminal activities, as defined by some legal framework. But it could also lead to plain censorship on political basis.
The TPM module is one of the most peculiar and controversial features of "trustworthy computing" systems. In fact, according to the specifications, the unique cryptographic key pair of each TPM is generated and stored during its production. Afterwards, it's kept inside the module and never communicated outside. This would make the root key impossible to modify, or even to retrieve, by everyone including the legitimate owner of the system. However, it is difficult to verify that each practical hardware realization, which may in fact differ significantly from others, is not subject to security problems, or is not part of a process where private data is nevertheless memorized or shared with other entities. This way, hardware manufacturers, or eventually state authorities, may acquire control of the root key, thus possibly compromising a user's device, including applications, data and online identities. Even if the root key is not leaked, however, there are other risks about users' online privacy. Some of these concerns have been mitigated in recent specifications of the TCG, which include the possibility to rely on a trusted third party or to use Direct Anonymous Attestation (DAA), with zero-knowledge proofs.
In a system which integrates a TPM module, the root key can be used for signing, encrypting or decrypting some data. However, since the key never leaves the TPM module, such data has to be passed to the TPM module. Moreover, the operation will be performed only under certain circumstances. In particular it has to be managed by a trusted application and resulting data has not to be stored outside of curtained memory. This way, data is inaccessible to other code, possibly including some parts of the operating system. Since the functionalities of the TPM module cannot be altered by anyone, including the owner of the system, some analysts have argued that, this way, the owner would be deprived of the ultimate control over its system (Schoen, 2003). An "Owner Override" feature has been proposed, to solve this problem. When imposing the owner override mode, the TPM would still protect its keys, but it would allow to produce signature, encryption and decryption of arbitrary data, thus empowering the user to produce false attestations. While allowing users to hold ultimate control over their systems, the "Owner Override" feature would make the whole architecture much less effective for DRM purposes.
DRM is one of the main possible application areas of trusted computing. It is not directly provided by a Trusted Computing platform, but it can be obtained by using its basic features, in particular attestation, curtained memory and sealed cryptographic key. In fact, DRM protected files can be encrypted in such a way that only trusted applications can access them, after attestation of full operation of the TPM, without alteration of the running code. In this case, some protected usage of the decryption key could be allowed for the trusted code. Various protection schemes and policies can be implemented, on the basis of the same feature set. However, the strict enforcement of copyright policies can limit also lawful behaviours, like those covered by "fair use" or "public interest" exemptions.
Given the cryptographic capabilities of the TPM module, it can also be used to issue attestations to certify that the computer has not been manumitted, according to the definition of system producers and data owners, and in this sense it is in a secure state. Attestation could be useful in corporate environments and large networks of computers, to control access to services made available on the network; in this case, only unmodified nodes would be granted access to sensible data and services. But this feature would be useful in network applications in general. Balfe et al. (2005) use TCG protocols

for Direct Anonymous Attestation (DAA) to enforce the stable use of pseudonyms in P2P networks, thus reducing pseudospoofing. As another example, multiplayer online games could be secured from most cheating attempts, like alteration of network traffic, application code, drivers or other system components. However, it could also open the door to anticompetitive behaviours, such as enforcing the use of a particular device, environment, or client application to interact with a remote server or participate into a distributed protocol.
In fact, one way of overcoming the resistance of users, would be the provision of premium content and services to "trusted" users, who connect though attestable trusted systems. Moreover, the exclusion from online communities and social activities could prove more effective than coercion, and even more effective in the area of entertainment devices and other specialized systems, where the user's control is already quite limited (Stajano, 2003), as in the case of Apple iPod and iPhone, Amazon Kindle etc.

TRUST MANAGEMENT

In contrast with the traditional approach to system security and also with new Trusted Computing architectures, based on Certification Authorities as trusted third parties, other solutions are possible. In Trust Management (TM) systems, in particular, the manager of local resources is considered as the ultimate source of trust about them, and it is provided with means to carefully administer the flow of delegated permissions. No a-priori trusted parties are supposed to exist in the system, in general, as this would imply some "obliged choice" of trust for the user, and without choice there is no real trust.
A number of architectures have been proposed for TM, including the Simple Distributed Security Infrastructure (SDSI) introduced by Rivest and Lampson (1996), and the Simple Public Key Infrastructure (SPKI) introduced by Ellison et al. (1999). They start from the observation that what computer applications often need is not to get the real-life identity of keyholders, but to make decisions about them as users (e.g. to grant access to a protected resource or not). More appropriately, Trust Management systems focus on principals and authorization. In general, a principal is any entity that can be taken accountable for its own actions in the system. In systems relying on asymmetric cryptography, principals could also be said to "be" public keys, i.e., if each principal has its own public key, then the principal can be identified directly through its own public key and rights to access system resources can be bound to the same key.
In a typical TM scheme, local names defined by a principal can be used on a global scale, if they are prefixed with the public key (i.e. the principal) defining them (Rivest and Lampson, 1996). Then, each principal can issue a Name Certificate to associate some name (in the issuer's namespace) with its intended meaning (either a public key or another name). A Name Certificate creates a name→subject bound and is defined as a 4-tuple: (issuer, name, subject, validity). There's no limitation to the number of keys which can be made valid meanings for a name. So in the end, a Name Certificate can be used to define a named group of principals. Li, Grosof and Feigenbaum (2003) interpret these named groups of principals as distributed roles, paving the way for a dRBAC (distributed Role-based Access Control) paradigm. In TM systems, the decision to delegate a role to some other entity is based uniquely on a local notion of trust, in principle based on socio-cognitive considerations (Falcone & Castelfranchi, 2001). This contrast with DRM and TC, where system manufacturers and other remote entities are entitled to define restrictions about the user's local operating environment, in terms of running software and permitted actions.
About dRBAC in general, Freudenthal et al. (2002) argue for the addition of some new features to previous approaches:

- Third-Party delegations allow some entities to delegate roles in different namespaces. This mechanism, related to a "speaks for" relationship, does not add any new functionality, as the same results can be obtained using anonymous intermediate roles, but improves the expressiveness and manageability of the system.

- Valued attributes allow to add attributes and corresponding numeric values to roles. This way, access rights for sensitive resources can be modulated according to some attributes. The same result could be obtained by defining different roles for different levels of access rights, but this would multiply the number of needed roles.
- Continuous monitoring allows to verify the actuality of trust relationships. Typically, this feature is based on a publish/subscribe protocol to advertise the status updates of relevant credentials, which can be either revocable or short-lived.

Another basic concept of TM systems is the Authorization Certificate, meant to create a straight authorization→subject bound (Ellison et al., 1999). It is defined as a 5-tuple: (issuer, subject, authorization, delegation, validity). Through an Authorization Certificate, a manager of some resources can delegate a set of access rights to a trusted subject. On its turn, this newly empowered principal can issue other certificates, granting a subset of its access rights to other entities. When finally requesting access to a resource, the whole certificate chain must be presented.

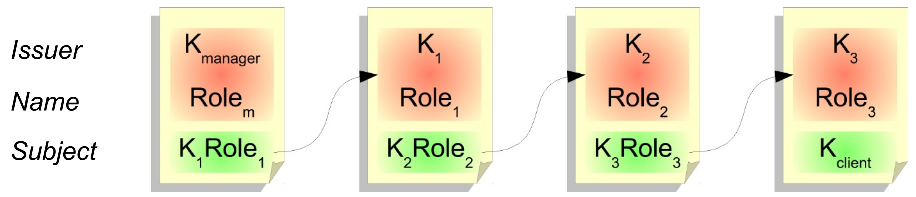

Figure 1. A delegation chain: $Role_m$ is granted indirectly from $K_{manager}$ to K_{client}

Li, Grosof and Feigenbaum (2003) discuss the importance of Authorization Certificates. Even recognizing that Authorization Certificates can improve the flexibility and granularity in permission handling, authors argue that most use cases can be satisfied by using local names and Name Certificates, only. In their perspective, local names are the distributed counterpart of roles in Role Based Access Control (RBAC) frameworks. Figure 1 shows how Name Certificates can be organized in a chain to link local roles, for delegation of access rights. Like roles, local names can be used as a level of indirection between principals and permissions. Both a local name and a role represent at the same time a set of principals, as well as a set of permissions granted to those principals. But, while roles are usually defined in a centralized fashion by a system administrator, local names, instead, are fully decentralized. This way, they better scale to Internet-wide peer-to-peer applications.

Automated Trust Negotiation

While Trust Management efforts are related to the protection of local resources, it has to be noted that credentials may convey sensitive data, as well. This data, including attributes, names and roles, needs to be protected at the same manner as other resources. Automated Trust Negotiation (ATN) (Winsborough and Li, 2000), in fact, enables interacting parties to build trust incrementally, possibly starting as strangers and reaching some level of trust by disclosing credentials and policies iteratively. This way of building trust allows entities to share resources or participate in a common protocol, after mutual acknowledgement, without relaying on third parties or on the enforcement of remote policies on local computing devices, like is required by TC and DRM systems. In this case, however, if a resource is transferred, there are no technical means to prevent further diffusion or some particular use (i.e. printing, or realizing a backup copy). If some behaviours are not allowed by contract, the issue is left on a legal level.
Various negotiation strategies are possible (Winsborough and Li, 2000), ranging from an eager strategy, in which credentials are disclosed as soon as it is allowed by the access control policy, to a parsimonious strategy, in which credentials are disclosed only after ensuring that a successful result will be reached. Yu, Winslett and Seamons (2003) describe the Disclosure Tree as a family of

strategies, which are proven to be interoperable with each other, i.e. interacting parties can choose each one a different strategy in the family and participate together to a negotiation.
Like credentials, policies may convey sensitive data and thus may need protection, too (Seamons, Winslett and Yu, 2001; Yu and Winslett, 2003). TrustBuilder is a modular system which can be used to perform customizable trust negotiation protocols (Winslett et al., 2002). Guidelines for the integration of TrustBuilder into a Service Oriented Architecture are discussed by Lee and Winslett (2008), in particular for realizing a Security Token Service which can carry on a Trust Negotiation conforming to WS-Trust specifications.
The various open issues of ATN, related to the protection of credentials and possible stall situations in the protocols, are addressed by a number of cryptographic protocols. For example, protocols for solving cyclic dependencies in disclosure policies include Oblivious Signature Based Envelopes (Li, Du and Boneh, 2005), Hidden Credentials (Bradshaw, Holt and Seamons, 2004), Oblivious Commitment Based Envelopes (Li and Li, 2005), and secret handshakes (Balfanz et al., 2003). Other protocols, focused instead on the separation of credential disclosure from attribute disclosure, include Private Credentials (Brands, 2000), Anonymous Credentials (Belenkiy et al., 2009), and OACerts (Li and Li, 2005).

Oblivious Attribute Certificates

In particular, to avoid the disclosure of sensitive information and help solving possible deadlocks during a Trust Negotiation, Li and Li (2005) present a family of protocols based on the concept of Oblivious Attribute Certificate (OACert). Instead of storing a user's attribute values in the clear, an OACert stores a cryptographic commitment of those attributes. Thus, the certificate alone does not allow anyone to gather information about the attribute values. Nevertheless, in a typical zero-knowledge scheme, OACerts may allow a certificate holder to satisfy some policy, disclosing some certificates but not the sensitive information associated with its attributes.
The scenario comprises three types of entities: (*i*) a trusted party T, responsible for certifying some attributes; (*ii*) a sender S, which is also a certificate holder; and (*iii*) a receiver R, which may be a service provider. The trusted party T is not necessarily bound to a hierarchical environment, as a X.509 Certification Authority, but it simply identifies an entity capable of issuing a certificate. An OACert is an assertion about the certificate holder, digitally signed by some trusted party. Each OACert contains one or more attributes. When the commitment system is secure, the certificate does not leak any information about sensitive attribute values. Since the protocol guarantees the separation of credential disclosure from attribute disclosure, the content of an OACert can be made public. The sender can show its OACert without having to worry about the privacy of its attributes. The generic scenario proceeds as follows: (*i*) each involved party owns a unique public/private key pair; (*ii*) a trusted party T generates an OACert for its future holder S; (*iii*) a receiver R, when presented with a request from a sender S, performs an access control on the basis of the attributes of S, certified in its OACert.
An attribute value in an OACert can be used in different ways to:

1. open the commitment and thus reveal the attribute value;
2. prove that an attribute value satisfies a condition, using a Zero-Knowledge Proof protocol and without revealing more information;
3. guarantee that the requester obtains a resource (e.g. a media file) only when its attribute values satisfy an access control policy; in this case, the Oblivious Commitment Based Envelope (OCBE) protocol is used and the operation happens without revealing anything about the attribute values, not even about the effective satisfaction of the policy.

EXPERIMENTATION WITH DELEGATION MECHANISMS FOR WEB SERVICES

This section will discuss the major issues faced for the realization of generic security mechanisms for

Web services, based on peer-to-peer delegation. In particular, it will present dDelega, a generic security library distributed as open source software (available at https://github.com/tomamic/dDelega). It allows issuing and verifying chains of delegation certificates. This allows a particular request for a service to be eventually associated with some roles and permissions. At its core, it defines an abstract Certificate class, extended by concrete classes representing Name Certificates, Authorization Certificates and Oblivious Attribute Certificates. Certificates are encoded as SAML assertions, with the possible inclusion of XACML policies, as these languages are expressive, flexible and extensible. SAML and XACML are readily integrated into a Service Oriented Architecture, yet they may serve in different application scenarios. In fact, dDelega is the result of ongoing work started with the development of a security layer for JADE, one the most widespread FIPA-compliant multi-agent systems (Poggi, Tomaiuolo and Vitaglione, 2005).

dRBAC

In TM systems based on asymmetric cryptography, principals can be identified directly by their public key. This may also be obtained in SAML. In fact, being designed to allow interoperability among very different security systems, SAML offers a variety of schemes for creating security assertions. In particular, there are a number of possible ways to represent a subject, including a SubjectConfirmation object to represent a subject as the holder of a certain public key (which is a principal in a TM system). One of the main aims of dDelega is to implement a distributed RBAC access control system, along the lines discussed in (Li, 2000). For this purpose, local names are particularly important, as they allow each principal to manage its own name space, which, on the other hand, is also one of the foundations of “federated identity” and SAML. In fact, while SAML allows the use of X.509 distinguished names, it also support a number of other heterogeneous naming schemes. In this sense, its reliance on XML provides intrinsic extendibility through schemas and namespaces.
In dDelega, assigning a local name to a public key, or to a set of public keys, is as simple as defining a role through a SAML assertion. In fact in SAML names and roles are not considered globally unique by design. And also assigning a named principal to a local name, or to a role, is perfectly possible. In particular, though not being foreseen in the specifications, it is perfectly possible to organize some SAML assertions into a certificate chain, like the one shown in Figure 1.
Ellison et al. (1999) note that, according to the X.509 PKI model, the issuer has the ability to eventually decide the conditions under which the certificate must be considered valid, and the enabled uses of the public key. But while the issuer must certainly be able to define the limits of its delegation, it is the final user who definitely takes a risk by accepting the certificate. Thus, if the relying party has to place some confidence in the certificate, it may need additional information about the assertion itself. In fact SAML allows the authentication authority to specify which mechanisms, protocols, and processes were used for the authentication.

Fine grained delegation

Thinking about the use of SAML as a representation of an Authorization Certificate, it would be important to have access rights, or permissions, associated with the subject. In dDelega, this is achieved through the integration of an XACML policy into a SAML assertion. The precise way to accomplish this is described in a separate profile of the standard (Anderson and Lockhart, 2004). From the Trust Management perspective, the conjunction of SAML and XACML, in particular the inclusion of XACML policies and authorization decisions into SAML assertions, provides a powerful tool for the delegation of access rights.
From this point of view, the fact that logic foundations of the XACML language exist is very important, as they provide XACML with a clear semantic. The problem is to find algorithms through which the combination of permissions granted in a chain of certificates could be computed in a deterministic way, as it is already possible in TM. In fact, even if the semantic of a XACML policy is logically sound,

nevertheless subtle problems can appear when different policies, linked in a chain of delegation assertions, have to be merged. One major problem is about monotonicity of authorization assertions, which cannot be guaranteed in the general case. Using XACML authorization decisions as SAML assertions, it is possible to assert that access to a particular resource is denied, instead of allowed. Though being a perfectly legal and meaningful concept, the denial of a permission (a “negative permission”) is not desirable in decentralized environments. In this case, a service provider can never allow access, as it cannot be sure to possess all issued statements. On the other hand, the non-monotonicity of the system can also lead to attacks, as issued assertions can be prevented to reach the provider, this way leading it to take wrong authorization decisions. Therefore, it is necessary to define a specific profile of SAML and XACML which could enable the secure delegation of permissions in decentralized environments, especially dealing with the case of “negative permissions”.

Oblivious attribute certificates

While introducing support for Oblivious Attribute Certificates inside dDelega, the set of APIs has been kept homogeneous with the rest of the library and the whole Java environment. In particular, the package for OACerts, just like the other packages of dDelega, can be used in different contexts, including applications not based on Web services. OACerts are implemented as a subclass of the dDelega Certicate class, and represented as SAML assertions.
With respect to implemented protocols, their organization has been kept as simple as possible, to guarantee easy of use and expansion. The protocols have been divided in two categories, according to their scope. The first group, descending from the VerifyScheme class, is meant to be used by a receiver, when it needs to match an oblivious credential against an access policy, before disclosing a resource. Two sublasses implement the DirectShow and ZeroKnowledge protocol. The second group, descending from the DecryptScheme class, allows (*i*) a provider to encrypt a resource to be sent to a requester; and (*ii*) the requester to decrypt the resource, if the access policy is satisfied. The initial implementation provides support for the EQ-OCBE (=) and GE-OCBE (≥) protocols.

Basic experimentation setting

After implementing the generic dDelega library, some experimentations have been conducted with simple Web services, exploiting its security mechanisms based on peer-to-peer delegation. In particular, a SOAP based Security Token Service (STS) has been developed. It is responsible for creating a security session valid on a whole platform, so that a client can send his selected certificates just once, and then access more services provided on that platform. A quite similar service has also been developed according to the RESTful paradigm. A different STS supporting Automated Trust Negotiation (ATN) is also being implemented, by integrating the TrustBuilder framework with a different rule engine and some configurable strategies.

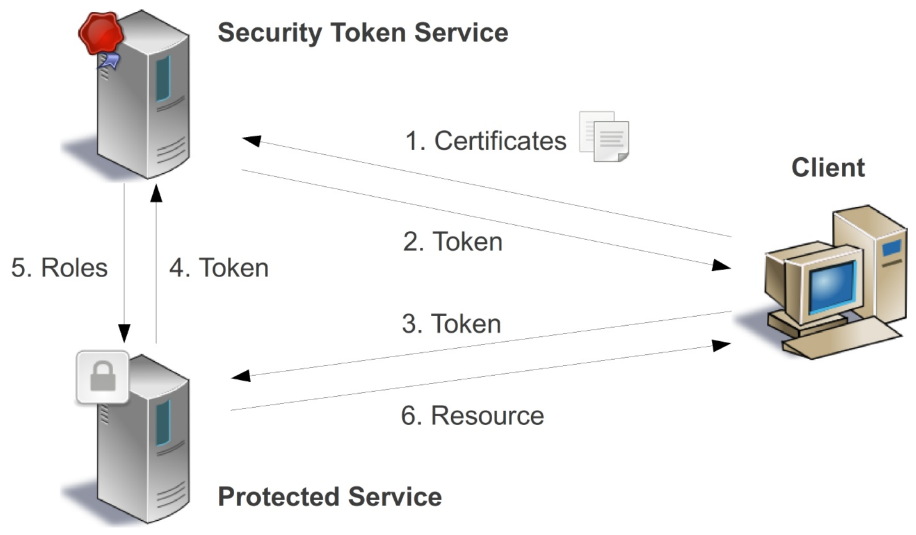

Figure 2. Experimentation scenario

The realized system is shown in Figure 2. It includes: (*i*) a Security Token Service (STS); (*ii*) a protected service, which needs proper authorization to be accessed; and (*iii*) a prototype client.
The STS is designed as a reusable service supporting the generic delegation of access rights among some peers. It allows a client to avoid attaching the same certificate chain to multiple service requests. The STS can accept and verify a certificate chain attached to a signed registration request. After a successful registration at the STS, the client is associated with a security session, which it can then specify when trying to access services on the same platform. The session token has to be obtained according to the WS-Trust specifications and it has to be used as a meaningful security token to be associated with WS-Security compliant messages.
The WS-Trust standard defines mechanisms for mediating trust relations among entities in the context of Web Services. It considers a security model in which a Web service can request that a received message proves a set of claims (e.g. name, key, privileges, etc) or, more commonly, that it carries a security token representing a relation between the sender and some other entity, trusted by the service provider. In this context, a service provider can request a client, before accessing its services, to present a token released by a trusted entity. A new client would probably not possess a proper token to access the service, in advance. For this reason, WS-Trust defines a protocol for allowing a client to contact an authority, trusted by the service provider, to request the token. Such an authority is in fact defined as a Security Token Service (STS). An STS, on his turn, can define the requirements which clients have to satisfy to obtain the release of a token. As a STS is responsible for releasing those tokens, it is also known as a “token issuer”.
The realized system effectively uses a number of technologies which have already been tested, and can work together to realize more complex scenarios than the ones foreseen in their specifications. In fact, the implementation is based on the Axis framework. Moreover all parties leverage Rampart to generate signed SOAP messages conforming to WS-Security specifications. All three parties are provided with their own couple of private and public keys, and can manage chains of delegation certificates and Oblivious Attribute Certificates encoded as SAML assertions.
The STS uses the dDelega library for verifying a chain of delegation certificates or some OACerts. To effectively handle a registration request, it also verifies the signature of the request message and, if the certificate chain is correct, eventually generates a security session and returns a proper token to the client. The token can only be used in association with the public key used to verify the request message. The presence of the STS is made explicit through the security policy of the protected service, in conformance to WS-Policy specifications
The protected service verifies the proper authorization before granting access. For this purpose, it is

associated with an Axis Handler which manages the session abstraction. The handler allows to separate the security aspect from the provision of the real service. Under the hood, it contacts the STS to receive a list of distributed roles associated with the public key and session token of the client, and then it uses an XACML policy to verify the association of the roles with the required permissions.
Finally, the client is built as an example and illustrates all the steps that a user application has to complete, to use the delegation mechanism.

Trust negotiation with TrustBuilder and Drools

A Security Token Service (STS), as defined by WS-Trust, is normall integrated into a system using a single round of messages, i.e. a RequestSecurityToken (RST), sent from the requestor to the STS, followed by a RequestSecurityTokenResponse (RSTR), sent from the STS to the requestor. However, in some scenarios, more steps may be needed before a token is obtained. In fact, the WS-Trust standard foresees the extension of this basic mechanism, named “negotiation and challenge framework”. The message exchange starts with a RST for requesting the token, then an arbitrary number of RSTR messages can be exchanged between the Requestor, or other entities, and the STS. Those RSTR messages may convey any additional information needed for completing the transaction, before finally transmitting the token. However, the WS-Trust standard does not further specify how to perform such a transaction. Thus, a generic protocol for ATN was defined, using some of the elements already proposed in (Lee and Winslett, 2008), when possible. However, the protocol and the content schemas are organized to better distinguish the two fundamental phases of the negotiation: (*i*) the initialization, and (*ii*) the real exchange of credentials.
A fundamental component of the system is the trust engine. It is responsible for evaluating which policies and credentials have to be inserted into the message at each round of the negotiation, on the basis of current state of negotiation and policies and credentials received at the previous round. In particular, TrustBuilder2 (TB2) was integrated into the system. It is a framework for trust negotiation, developed for providing a flexible and extensible tool in the context of research about this problem area. TrustBuilder2 has not been realized for usage in the context of Web services, however his modular structure allows it to be extended for: (*i*) using different policy languages, (*ii*) implement different negotiation strategies, (*iii*) and provide support for different types of credentials. In particular, after a proper translation we defined, it is able to evaluate policies expressed according to the WS-SecurityPolicy language. Moreover, for being integrated into dDelega, TrustBuilder2 has been customized and extended for using a different rule engine as a policy checker. In particular, the Drools rule engine was used for the policy checker component instead of Jess, supported by the currently available version of TB2. Drools is based on the so-called ReteOO algorithm, i.e., an adaptation of the Rete algorithm for object oriented systems.
Including the initialization phase and a request for credentials from both sides, the whole process defined in our test takes 4 rounds, in which both the client and the STS send a message to the other party. Without optimizations, the execution times vary around a mean value of 6 seconds, including the signature and encryption of SOAP messages, and 4.5-5 seconds without any signature and encryption. Thus, the execution of an ATN process proved to be functional but probably quite costly in many scenarios. Thus, it is advisable to release tokens which can be used for accessing a number of cohesive services in a given time interval, without repeating the negotiation.

CONCLUSION

This article have analysed the basic architectural principles and functioning of Digital Rights Management and Trusted Computing, on the one hand, and decentralized Trust Management, on the other hand.
The basic application of DRM and TM may appear similar, since both technologies deal with access rights. However, TM sprung from the observation that a global centralized or hierarchical Public Key

Infrastructure (PKI) was both impractical and undesirable, and has shown that the flow of trust and access rights can be driven directly by the community of service providers and consumers, in a peer to peer fashion. Trust Management systems allow peers to grant trust to each other for accessing local resources, and to create delegation chains based on the local evaluation of trust as a quantifiable socio-cognitive element.

DRM and TC, instead, deal with the enforcement on a computing device of access policies defined not only by its user, but also by system manufacturers and copyright holders. Stallman (2002) and other critics describe these “digital rights” as “restrictions”, since they arguably limit the operations which a user can perform on his own computer or device. Basically, they assure copyright holders and media producers that the hosting system will respect the access restrictions they defined. To this aim, they need a newly founded global PKI, with even more inner complexity, since the number of roles is increased, yet the trust relations among them are still quite vague.

In fact, TM and TC systems are founded on contrasting visions, rooted at the semantics of trust and the control of trust flow. Their different interpretation of trust drives to divergent consequences from the points of view of system architecture, access policies, law, ethics, politics. It can be noted, however, that from the technological point of view those systems are quite orthogonal and in some sense complementary. In principle, the two systems can be combined. This is quite obvious, since the peer-to-peer nature of TM systems allows users to assign trust to any entity, without excluding centralized or hierarchical ones. However, in this case the principles of Trust Management would be virtually nullified, since some entities are conceded the central position in the administration of the operations which the local user is entitled to perform. Moreover, the enforcement of remote access control policies on a device, for restricting its user's operations, is in contrast with the complete control over owned local resources assured by TM systems.

REFERENCES

Arbaugh, B. (2002). Improving the TCPA specification, *IEEE Computer* 35(8), 77–79.

Arbaugh W.A., Farber, D.J., and Smith, J.M. (1997). A secure and reliable bootstrap architecture, *Proceedings of the 1997 IEEE Symposium on Security and Privacy (S&P 1997)* (Oakland, California, USA), IEEE Computer Society Press, Los Alamitos, California, May 1997, pp. 65–71.

R. Anderson, R. (2003). Cryptography and competition policy — Issues with ‘trusted computing’, *Proceedings of PODC ’03*, July 13-16, 2003, Boston, Massachsetts, USA, ACM, 2003, pp. 3–10.

Anderson, R. (2003). *Trusted Computing Frequently Asked Questions. Version 1.1*. Retrieved 2013-08-05 from http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html

Anderson, A., and Lockhart, H. (2004). *SAML 2.0 profile of XACML*. Retrieved 2013-08-05 from http://docs.oasis-open.org/xacml/access_control-xacml-2.0-saml_profile-spec-cd-02.pdf

Balacheff, B., Chen, L., Pearson, S., Proudler, G., and Chan, D. (2000). Computing platform security in cyberspace, *Information Security Technical Report* 5(1), 54–63.

Balfanz, D., Durfee, G., Shankar, N., Smetters, D., Staddon, J., and Wong H. C. (2003). Secret Handshakes from Pairing-Based Key Agreements. In *IEEE Symposium on Security and Privacy*, pp. 180-196.

Balfe, S., Lakhani, A. D., & Paterson, K. G. (2005, August). Trusted computing: Providing security for peer-to-peer networks. In *Peer-to-Peer Computing, 2005. P2P 2005. Fifth IEEE International Conference on* (pp. 117-124). IEEE.

Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., and Shacham, H. (2009). Randomizable Proofs and Delegatable Anonymous Credentials. In Advances in Cryptology - CRYPTO 2009, ser. *Lecture Notes in Computer Science* vol. 5677, Springer Berlin / Heidelberg, pp. 108-125.

Bertino E., Squicciarini, A. C., Paloscia, I., and Martino, L. (2006). Ws-AC: a fine grained access control system for web services. *World Wide Web*, 9(2), 143-171.

Bhargavan, K., Fournet, C., Gordon, A.D., and Corin, R. (2007). Secure sessions for web services. *ACM Transactions on Information and System Security*, 10(12).

Bhatti, R., Joshi, J. B. D., Bertino, E., and Ghafoor, A. (2003). Access Control in Dynamic XML-based Web-Services with XRBAC. In *Proceedings of the First International Conference on Web Services*, Las Vegas.

Bradshaw, R. W., Holt, J. E., and Seamons, K. E. (2004). Concealing complex policies with hidden credentials. In *Proceedings of the 11th ACM conference on Computer and communications security (CCS'04)*, pp. 146-157.

Brands, S.A. (2000). *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. ISBN:9780262024914. MIT Press.

Bright, P. (2013). *DRM in HTML5 is a victory for the open Web, not a defeat*. Ars Technica, 2013-05-10. Retrieved 2013-08-05 from http://arstechnica.com/business/2013/05/drm-in-html5-is-a-victory-for-the-open-web-not-a-defeat/

Bussard, L., Nano, A., and Pinsdorf, U. (2009). Delegation of access rights in multi-domain service compositions. *IDIS Journal (Identity in the Information Society)*, Springer Netherlands, 2(2), 137-154.

Cantor, S. (2005). *Shibboleth Architecture. Protocols and Profiles*. Retrieved 2013-08-05 from http://shibboleth.internet2.edu/shibboleth-documents.html

Carrier, M. A. (2013). SOPA, PIPA, ACTA, TPP: An Alphabet Soup of Innovation-Stifling Copyright Legislation and Agreements. *Nw. J. Tech. & Intell. Prop.*, *11*, 21-163.

Chadwick, D. W., Zhao, G., Otenko, S., Laborde, R., Su, L., and Nguyen, T. A. (2008). PERMIS: a modular authorization infrastructure. *Concurrency and Computation: Practice and Experience*, 20(11), 1341-1357.

Chen, L., Pearson, S., Proudler, G., Chan, D., and Balacheff, B. (2000). How can you trust a computing platform?. *Proceedings of Information Security Solutions Europe (ISSE 2000)*.

Clark, P. C., & Hoffman, L. J. (1994). BITS: a smartcard protected operating system. *Communications of the ACM*, *37*(11), 66-70.

Coyle, K. (2003). *The Technology of Rights: Digital Rights Management*. Retrieved 2013-08-05 from http://www.kcoyle.net/drm_basics.pdf

DPRL (1998). *The Digital Property Rights Language, Manual and Tutorial – XML Edition, Version 2.00*. November 13, 1998. Retrieved 2013-08-05 from http://xml.coverpages.org/DPRLmanual-XML2.html

Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., and Ylonen, T. (1999). SPKI certificate theory. *IETF RFC 2693*, September 1999.

England, P., Lampson, B., Manferdelli, J., & Willman, B. (2003). A trusted open platform. *Computer*, *36*(7), 55-62.

England, P., & Peinado, M. (2002, January). Authenticated operation of open computing devices. In *Information Security and Privacy* (pp. 346-361). Springer Berlin Heidelberg.

Falcone, R., & Castelfranchi, C. (2001, May). Social trust: a cognitive approach. In *Trust and deception in virtual societies* (pp. 55-90). Kluwer Academic Publishers.

Freudenthal, E., Pesin, T., Port, L., Keenan, E., and Karamcheti, V. (2002). dRBAC: Distributed Role-based Access Control for Dynamic Coalition Environments. In *Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems (ICDCS'02)*, 2002, pp. 411-420.

Garfinkel, T., Rosenblum, M., & Boneh, D. (2003, May). Flexible OS Support and Applications for Trusted Computing. In *HotOS* (pp. 145-150).

Greene, J. (2012). *Intel Trusted Execution Technology, white paper*. Retrieved 2013-08-05 from http://www.intel.com/txt/

Halpin, H. (2013). *DRM and HTML5: it's now or never for the Open Web*. The Guardian, 2013-06-06. Retrieved 2013-08-05 from http://www.theguardian.com/technology/2013/jun/06/html5-drm-w3c-open-web

Iannella, R. (2002). Open Digital Rights Language (ODRL), Version: 1.1. Retrieved 2013-08-05 from http://odrl.net/1.1/ODRL-11.pdf

Itoi, N., Arbaugh, W. A., Pollack, S. J., & Reeves, D. M. (2001, January). Personal secure booting. In *Information Security and Privacy* (pp. 130-144). Springer Berlin Heidelberg.

Kuhlmann, D., Landfermann, R., Ramasamy, H., Schunter, M., Ramunno, G., & Vernizzi, D. (2006). *An open trusted computing architecture—secure virtual machines enabling user-defined policy enforcement*. Retrieved 2013-08-05 from http://www.opentc.net/

Lee, A. J., and Winslett, M. (2008). Towards Standards-Compliant Trust Negotiation for Web Services. In *Proceedings of the Joint iTrust and PST Conferences on Privacy, Trust Management, and Security (IFIPTM 2008)*.

Li, N. (2000). Local names in SPKI/SDSI. In *Proceedings of the 13th IEEE Computer Security Foundations Workshop*, IEEE Computer Society Press, pp. 2-15.

Li, N., Du, W., and Boneh, D. (2005). Oblivious Signature-Based Envelope. *Distributed Computing*, 17(4), pp. 293-302.

Li, N., Grosof, B. N. and Feigenbaum, J. (2003). Delegation logic: a logic-based approach to distributed authorization, *ACM Transactions on Information and System Security*, 6(1), 128-171.

Li, J., and Li, N. (2005). OACerts: Oblivious attribute certificates. In Proceedings of the 3rd Conference on Applied Cryptography and Network Security (ACNS), ser. *Lecture Notes in Computer Science* vol. 353, pp. 301-3017. Springer.

O'Riordan, C. (2006). Transcript of Opening session of first international GPLv3 conference. January 2006. Retrieved 2013-08-05 from http://www.ifso.ie/documents/gplv3-launch-2006-01-16.html

Pearson, S., & Balacheff, B. (Eds.). (2003). *Trusted computing platforms: TCPA technology in context*. Prentice Hall Professional.

Pearson, S. (2002). *Trusted computing platforms, the next security solution*. HP Labs.

Pedersen, T. (1991). Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In Advances in Cryptology — CRYPTO '91, ser. *Lecture Notes in Computer Science* vol. 576, Springer Berlin / Heidelberg, pp. 129-140.

Peinado, M., Chen, Y., England, P., & Manferdelli, J. (2004, January). NGSCB: A trusted open system. In *Information Security and Privacy* (pp. 86-97). Springer Berlin Heidelberg.

Pfitzmann, B., Riordan, J., Stüble, C., Waidner, M., & Weber, A. (2001, April). The PERSEUS system architecture. In *VIS* (pp. 1-18).

Poggi, A., Tomaiuolo, M., and Vitaglione, G. (2005). A security infrastructure for Trust Management in Multi-Agent Systems. In Trusting Agents for Trusting Electronic Societies, ser. *Lecture Notes in Computer Science* vol. 3577, pp 162-179.

Reid, J., Nieto, J. G., Dawson, E., and Okamoto, E. (2003, September). Privacy and trusted computing. *Proceedings 14th International Workshop on Database and Expert Systems Applications, 2003*. (pp. 383-388). IEEE.

Rivest, R.L., and Lampson, B. (1996). *SDSI - A Simple Distributed Security Infrastructure*. September 15, 1996. http://people.csail.mit.edu/rivest/sdsi11.html (Accessed 20 August 2012).

Sadeghi, A. R., & Stüble, C. (2004). Taming "trusted platforms" by operating system design. In *Information Security Applications* (pp. 286-302). Springer Berlin Heidelberg.

Sadeghi, A. R., Stüble, C., & Pohlmann, N. (2004). European multilateral secure computing base. *Datenschutz und Datensicherheit*, 548-554.

Schneier, B. (2002). Crypto-Gram Newsletter August 15, 2002. Retrieved 2013-08-05 from http://www.schneier.com/crypto-gram-0208.html

Seamons, K.E., Winslett, M., and Yu, T. (2001). Limiting the disclosure of access control policies during automated trust negotiation. In *Proceedings of the Network and Distributed Systems Symposium*.

She, W., Thuraisingham, B., and Yen, I-L. (2007). Delegation-based security model for web services. In *Proceedings of 10th IEEE High Assurance Systems Engineering Symposium (HASE '07)*, IEEE Computer Society, ISBN:978-0-7695-3043-7, pp. 82-91.

Schoen, S.D. (2003). EOF - Give TCPA an Owner Override. Linux Journal 116, December 2003. Retrieved 2013-08-05 from http://www.linuxjournal.com/article/7055

Stallman, R. (2002). Can you trust your computer?. *Free Software, Free Society: Selected Essays of Richard M. Stallman*, 115-118.

Stallman, R. (1997). The right to read. *Communications of the ACM*, *40*(2), 85-87.

Stamp, M. (2003). Digital Rights Management: The Technology Behind The Hype. *Journal of Electronic Commerce Research*, 4(3), 102-112.

Tygar, J., and Yee, B. (1991). *Dyad: A system for using physically secure coprocessors*. Technical Report CMU-CS-91-140R, Carnigie Mellon University, Pittsburgh, Pennsylvania, USA, May 1991.

Welch, V., Foster, I., Kesselman, C., Mulmo, O., Pearlman, L., Tuecke, S., Gawor, J., Meder, S., and Siebenlist, F. (2004). X.509 Proxy Certificates for Dynamic Delegation. In *Proceedings of the 3rd Annual PKI R&D Workshop*, Gaithersburg MD, USA, NIST Technical Publications.

Welch, V., Siebenlist, F., Foster, I., Bresnahan, J., Czajkowski, K., Gawor, J., Kesselman, C., Meder, S., Pearlman, L., and Tuecke, S. (2003). Security for Grid services. In *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*, pp. 48-57.

Winsborough, W. H., and Li, N. (2000). Automated trust negotiation. In *DARPA Information Survivability Conference and Exposition*, volume I, pp. 88-102, IEEE Press.

Winslett, M., Yu, T., Seamons, K.E., Hess, A., Jacobson, J., Jarvis, R., Smith, B., and Yu, L. (2002). Negotiating trust in the Web. *IEEE Internet Computing*, 6(6), 30-37.

Walker, J. (2003). *The Digital imprimatur: How big brother and big media can put the Internet genie back in the bottle*. Retrieved 2013-08-05 from http://www.fourmilab.ch/documents/digital-imprimatur/

XrML (2002). *XrML 2.0 Technical Overview Version 1.0*. March 8, 2002. Retrieved 2013-08-05 from http://www.xrml.org/Reference/XrMLTechnicalOverviewV1.pdf

Yu, T., Winslett, M., and Seamons, K. E. (2003). Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. *ACM Transactions on Information and System Security*, 6(1), 1-42.