PLI: A New Framework to Protect Digital Content for P2P Networks

Guofei Gu[1], Bin B. Zhu[2], Shipeng Li[2], and Shiyong Zhang[1]

[1] Dept. of Computing & Info. Tech., Fudan Univ., Shanghai 200433, China
gu_guofei@yahoo.com, szhang@fudan.edu.cn
[2] Microsoft Research Asia, Beijing 100080, China
{binzhu,spli}@microsoft.com

Abstract. In this paper, we first propose a novel Public License Infrastructure (PLI) that uses cryptographic threshold secret sharing schemes to provide decentralized public license services for the Digital Rights Management (DRM). This distributed PLI replaces the centralized license server in a conventional DRM system. PLI offers many advantages such as intrusion and fault tolerance, flexibility, scalability, reliability, high availability. We then propose a PLI-based DRM system to provide content protection and digital rights management for users of Peer-to-Peer (P2P) networks. This DRM system is especially useful for small content providers such as peers in a P2P network who cannot afford the conventional server/client based DRM system and traditional distribution channels.

Keywords: Peer-to-Peer (P2P), digital content protection, Digital Rights Management (DRM), Public License Infrastructure (PLI), License Authority (LA), intrusion tolerance, secret sharing, proactive shares update, distributed trust model.

1 Introduction

Wide availability of digital content such as software and digital media and easy access through Internet have prompted the demand for technologies to protect digital content from illegal access or copy. Digital Rights Management (DRM) is a system which manages all rights for digital content from creation to consumption [1][2]. Most DRM systems such as Adobe's EBooks [3], Intertrust's [4], and Microsoft's [5] DRM systems are based on encryption. Digital content is encrypted and distributed. A consumer who wants to play the protected content will first get the access permission and the decryption key. The DRM system enforces the proper usage of the digital content. Typically the access rules or rights and the decryption key are contained in a encrypted license. In these systems, license acquisition is based on the classical server/client model in which a user acquires a license from a centralized and dedicated license server.

This work was done when Guofei Gu was a visiting student at the Microsoft Research Asia.

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 206–216, 2003.

Peer-to-peer (P2P) networks have recently attracted increasing attention in both academia and the industry. P2P networks offer many desirable features: adaptation, self-organization, load-balance, fault-tolerance, low cost, high availability, scalability, and a large pool of resources. P2P networks have emerged as a popular way to share huge amounts of data (e.g., [6][7]). Most P2P networks do not have any digital rights management or access control. P2P networks are often blamed for illegally sharing copyrighted materials.

In a conventional DRM system, all license acquisition requests are processed by a centralized license server. This makes the license server heavy-loaded, complex and expensive to run and maintain, and a weak link in the system. Failure of the license server disrupts normal DRM services. On the other hand, small content providers such as a peer in a P2P network may not afford the cost of services of the license server. In this paper, we leverage the P2P concept to propose a novel distributed infrastructure called the *Public License Infrastructure* (PLI) to provide decentralized license services for DRM. PLI consists of many trusted *License Authorities* (LAs) which collectively provide DRM license services for consumers. Secrets are shared among LAs with a threshold secret sharing scheme. PLI provides an inexpensive, reliable, and highly available alternative to the centralized license server in a conventional DRM system. Based on PLI and LAs, we then propose a DRM system which provides content protection and digital rights management for users of P2P networks. The proposed DRM system is especially useful for small content providers such as peers in a P2P network who cannot afford the conventional server/client based DRM system and traditional distribution channels. The proposed DRM system can be considered as a hybrid P2P network where some trusted nodes, i.e., LAs, play a role as mini-servers. To the best of our knowledge, our proposed system is the first distributed DRM license service system and the first DRM system designed for P2P networks.

The paper is organized as follows: Section 2 introduces the background and related work for the paper, which includes the conventional DRM system, P2P networks, and distributed certificate authority schemes. Section 3 presents the proposed public license infrastructure and the DRM system designed for P2P networks. It is followed with discussions in Section 4. We conclude the paper in Section 5.

2 Background

2.1 Conventional Digital Rights Management (DRM) Systems

Conventional DRM systems are based on the server/client model. A license server is used to provide license services for consumers. A typical example is the Microsoft Windows Media Rights Manager [5] which contains the following five processes: packaging, distribution, establishing a license server, license acquisition, and playing the content [8]. These processes are briefly described in the following:

i. **Packaging.** The rights manager encrypts the digital media and then packages the content into a digital media file. The decryption key is stored in an encrypted license which is distributed separately from the media file. Other

information such as a link to the license is added to the media file to facilitate license acquisition.

ii. **Distribution.** The packaged file is distributed to users through some distribution channels such as downloading, streaming, and CD/DVD. There is no restriction on distribution of the packaged content.

iii. **Establishing a license server.** The content provider (referred to as the publisher in the following) chooses a license clearing house that stores the specific rights or rules of the license and runs a license server which is used to authenticate the consumer's request for a license. Licenses and protected media files are distributed and stored separately to make it easier to manage the entire system.

iv. **License acquisition.** To play the protected content, a consumer first acquires a license which contains the decryption key and the rights the consumer has with the content. This process can be done in a transparent way to the consumer or with minimal involvement of the consumer (such as when payment or information is required).

v. **Playing the content.** A player that supports the DRM system is needed to play the protected content. The DRM system ensures that the content is consumed according to the rights or rules included inside the license. Licenses can have different rights, such as start times and dates, duration, and counted operations. Licenses, however, are typically not transferable. Each consumer has to acquire his or her own license to play the protected content.

2.2 Peer-to-Peer (P2P) Networks

With the introduction of the first P2P network several years ago, we have seen an explosive growth of P2P networks and their applications. Well-known P2P networks include Napster [9], Gnutella [10], JXTA [11], Freenet [12], Chord [13], CAN [14], Pastry [15], Tapestry [16], etc. P2P networks are also actively studied and developed by researchers in the fields.

P2P networks offer many desirable features such as redundancy and fault tolerance. Data gradually spreads and gets replicated at many nodes, and thus is highly redundant in a P2P network. High redundancy means high reliability and availability. This can effectively reduce the operational cost and serve more users. Despite these desirable features, P2P networks still lag behind the traditional server/client paradigm in security, efficiency, performance guarantees like atomicity and transactional semantics. P2P networks lack of content rights management and access control which threatens their healthy development and wide adoption. Companies building P2P software have been sued in courts by large content providers for illegal sharing of copyrighted materials the P2P software enables. The proposed DRM system in this paper provides a DRM system for P2P networks.

2.3 Threshold-Based Secret Sharing and Distributed Certificate Authorities

Threshold-based secret sharing [17][18] and proactive secret share updates [19][20] have been studied actively in cryptography. They are the basis for the proposed

system in this paper, as well as many other proposed schemes of distributed certificate authorities. The Intrusion Tolerance via Threshold Cryptography (ITTC) project at Stanford [21] enables a private RSA key to be shared among *k* servers such that any *k-1* or *k-2* of them can decrypt incoming messages without reconstructing the key. The Partially Distributed Certificate Authority (PDCA), an ad-hoc key management scheme proposed in [22], uses a (*k, m*) threshold scheme to distribute services of the certificate authority to a set of specialized server nodes. Each of these nodes is capable of generating a partial certificate using its share of the certificate signing key. A combination of *k* partial certificates can generate a valid certificate. The partial certificates are fixed, or more precisely, logically centralized. A specialized server is needed to combine the partial certificates. The Fully Distributed Certificate Authority (FDCA), another ad-hoc key management solution proposed in [23] and analyzed in [24][25], uses a (*k, m*) threshold scheme to distribute an RSA certificate signing key to all nodes in a network. It also uses verifiable and proactive secret sharing mechanisms to protect against denial of service attacks and attacks to compromise the certificate signing key. Unlike PDCA, FDCA distributes the service to all the nodes when they join the network. There is no need to elect or choose any specialized server nodes. This scheme, however, is based on the assumption that every node has a reliable intrusion and misbehavior detection function, which may be impractical in many applications.

3 Public License Infrastructure (PLI) and PLI-Based DRM

3.1 Public License Infrastructure (PLI) and License Authority (LA)

In a conventional server/client based DRM system, every publisher has to choose a license clearing house with a centralized license server to provide license services for consumers. Small publishers may not afford to do so and, as a consequence, their content will not be protected by the conventional DRM. This is especially true in a P2P network where every user can be both a consumer and a publisher. An inexpensive license service and DRM system is desirable to protect small publishers. The Public License Infrastructure (PLI) we propose here serves as an inexpensive license service provider. PLI is similar to the Public Key Infrastructure (PKI). Instead of signing certificates for public keys, PLI is a decentralized system to provide public license services for all the users in a DRM system. Like Certification Authorities (CAs) in PKI, we have many License Authorities (LAs) in PLI. These LAs are fully trusted in the system. To build PLI for a P2P network, we need to have enough trusted nodes as LAs. In this paper we assume that the underlying network always has enough number of trusted nodes, even when the underlying network is a P2P network.

To build distributed license services, we split a secret into *m* partial secrets and upload the partial secrets to LAs. A (*k, m*) threshold secret sharing scheme is used to split and recover the original secret. A LA can have more than one partial secret. In the extreme case, one LA can contain all the *m* partial shares. In this case, the LA behaves like the centralized license server in the conventional DRM system, even though the underlying licensing mechanisms are different. Details for using LAs for

issuing a license in the proposed DRM system will be given in subsequent subsections.

If a network supports PKI, PLI can be easily built on top of PKI. Each CA in PKI is assigned with additional task to store and maintain partial secrets and to provide public license services. In other words, each CA is extended in functions to be a LA too.

PLI is distributed and intrusion-tolerant. Even some LAs in a PLI are compromised, as long as the number of compromised partial shares is less than k, the system is still secure. This intrusion tolerance in PLI is similar to the fault tolerance provided by a P2P network. PLI is in fact a distributed trust architecture.

In the following, we shall describe the detail of the PLI based DRM system for P2P networks we propose in this paper.

3.2 Content Packaging and Distribution

Content packaging in our DRM system is the same as in the conventional DRM system which is described in Section 2.1. In this stage, a strong symmetric encryption algorithm such as the Advanced Encryption Standard (AES) [26] is used to encrypt the content.

Because our DRM system is built for P2P networks, content distribution is efficient and trivial. A publisher delivers the packaged content into a P2P network and the content is replicated to many nodes. A consumer can use the inherited search mechanism in a P2P network to easily locate and retrieve the desired content. The replication and cache mechanisms in a P2P network provide a good fault tolerance.

3.3 Establish Pre-license and Secret Sharing

A license contains the decryption key to unlock the content and some access rules or rights expressed in certain languages. Many well-known languages can be used to express rights, for example XRML (eXtensible Rights Markup Language) [27], XACML (eXtensible Access Control Markup Language) [28], ODRL (Open Digital Rights Language) [29], etc.

There are two types of licenses in our DRM system. The first type of license is called the *pre-license*, which is generated at the content packaging stage and will be used to generate the other type of license. The second type of license is called the *formal-license*, which is the license a consumer uses to play the protected content. The pre-license contains the decryption key associated with the access rules that the publisher of the content allows. The formal-license contains the decryption key and the access rules that a consumer, the owner of the formal-license, has. The formal-license will be described in the next subsection.

The pre-license, denoted as *prel* in the following, is encrypted with an asymmetric encryption algorithm such as RSA [30] and a public key *PK:*

$$prel = (license)^{PK}. \tag{1}$$

The corresponding secret private key *SK* is divided into *m* shares using a (*k, m*) threshold secret sharing scheme, which will be described in detail next. The publisher chooses *m* LAs and uploads one partial secret share to one chosen LA, along with the pre-license *prel* and the license ID. It is also possible to upload more than one secret share to a LA, or to choose more than *m* LAs and upload one secret share to multiple LAs. Recall that we have assumed that there exist trusted nodes in a network and these nodes operate as LAs for the network. Addresses of the partial secret holding LAs and other information are packaged with the content to facilitate a consumer to locate right LAs in generating the formal-license.

To obtain the *m* partial secrets, the following steps are performed:

i. The publisher generates the sharing polynomial

$$f(x) = a_0 + a_1 x + ... + a_{k-1} x^{k-1} \tag{2}$$

over a finite field Z_N where $a_0 = SK$.

ii. Each LA, identified by $id_i, i = 1, \cdots, m$, of the *m* chosen LAs is securely uploaded with the polynomial share

$$S_i = f(id_i) \mod N. \tag{3}$$

iii. The publisher broadcasts the *k* public witnesses of the sharing polynomial's coefficients $\{g^{a_0}, \cdots, g^{a_{k-1}}\}$, where $g \in Z_N$ after which it destroys the polynomial and quits.

iv. Each LA id_i verifies validity of the received share by checking if the following equation holds:

$$g^{S_i} = g^{a_0} \cdot (g^{a_1})^{id_i} \cdot \ldots \cdot (g^{a_{k-1}})^{id_i^{k-1}} \tag{4}$$

3.4 Formal-License Acquisition and Playing the Content

When a consumer retrieves the protected content and tries to play, the player checks whether a valid formal-license for the content is available at the local machine, and also checks if the content can be accessed. If these checks are OK, the player plays the content. Otherwise the player finds the LAs from the packaged content and contacts *k* live LAs for generating a formal-license for the consumer. The consumer might be involved for information registration or for payment which LAs can facilitate. After receiving the request, each LA responds with the partial result generated from its partial secret share. After receiving *k* correct partial results, the player combines them and generates a formal-license which typically binds to the specific machine the player runs. The player checks the access rules in the formal-license and plays the content. Typically all the DRM related operations at a local machine are performed by a black-box DRM module inside or coupled with the player. This module should be secure and tamper-proof.

Fig. 1 shows an example using a (*2, 3*) threshold secret sharing scheme with three LAs. In this case, a failure of one LA does not affect the proper function of the system.

The detail of generating the formal-license is described below:

i. The node *p* where the player runs gets the list of partial secret holding LAs and contacts *k* live LAs with the license ID and the formal-license acquisition request.

ii. Each LA id_i calculates the partial result $prel_i$ from its partial secret share S_i :

$$prel_i = (prel)^{S_i} \bmod N . \tag{5}$$

It generates a random number u and calculates $A_1 = g^u$, $A_2 = prel^u$, $r = u - c \cdot S_i$, and

$$c = hash\,(g^{S_i}, prel_i, A_1, A_2) . \tag{6}$$

The LA responds by sending $prel_i$, A_1, A_2, and r securely to the requesting node *p*.

iii. The node *p* calculates

$$g^{S_i} = g^{a_0} \cdot (g^{a_1})^{id_i} \cdot \ldots \cdot (g^{a_{k-1}})^{id_i^{k-1}} \tag{7}$$

from the public witnesses of the sharing polynomial's coefficients. Eq. (6) is then applied to g^{S_i} and the received $prel_i$, A_1, A_2 to calculate *c*. The received partial result $prel_i$ is verified by checking if the following equations hold: $g^r \cdot (g^{S_i})^c = A_1$ and $prel^r \cdot (prel_i)^c = A_2$. The above steps are repeated until the node *p* gets *k* valid partial results. If less than *k* valid partial results can be obtained, the formal-license generation fails.

iv. The node *p* uses the *k* valid partial results to calculate the license:

$$\begin{aligned} license &= \prod_i (prel_i)^{l_{id_i}(0)} = (prel)^{\sum_i S_i \cdot l_{id_t}(0)} \\ &= (prel)^{SK} = ((license)^{PK})^{SK}, \end{aligned} \tag{8}$$

where $l_{id_i}(x) = \prod_{j=1, j\neq i}^{k} \frac{x - id_j}{id_i - id_j}$. The access rules in the license may be modified to reflect the actual rights the consumer gets. The license is then encrypted with some secret key related to the specific hardware of the node *p* to generate a formal-license, an individualized license that can be used only by the machine. The formal-license is stored in the local machine for future access.

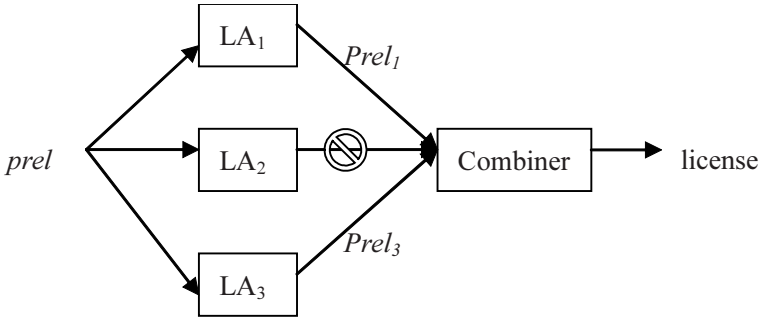

Fig. 1. An example with a (*2, 3*) threshold scheme where failure of one LA does not affect the normal operation.

3.5 Proactive Shares Update

In the secret sharing scheme described above the secret is protected by distributing partial secrets among LAs. Given sufficiently long time an attacker may finally compromise *k* partial secrets to deduce the secret key *SK*. To thwart such an attack, the partial secret shares are updated periodically with a proactive secret sharing scheme. An attacker Bob has to compromise *k* partial secrets before the partial secrets are updated. Otherwise he has to restart his attacks again. The proactive secret share update algorithms proposed in [18][19][20] can be applied to create a community of *m* entities with the new version of secret shares.

At periodic intervals the LAs update their shares of the private key *SK*. At the beginning of an update, each LA *i* generates a random *(k, m)* sharing of the secret *0* using a random update polynomial $f_{i,update}(x)$:

$$f_{i,update}(x) = b_{i,1}x + ... + b_{i,k-1}x^{k-1} \bmod N \tag{9}$$

Then LA *i* calculates the subshares $S_{i,j} = f_{i,update}(j)$, $j = 1, \cdots, m$, and distributes $S_{i,j}$ to all LA *j*, $j = 1, \cdots, m$.

Now each LA *i* has *m* subshares $S_{j,i}$, $j = 1, \cdots, m$ from all LAs. These subshares are added to the original share S_i, and the result is the new updated share:

$$S_i' = S_i + \sum_{j=1}^{m} S_{j,i} \ . \tag{10}$$

The corresponding new secret sharing polynomial $f_{new}(x)$ is the summation of the original polynomial $f(x)$ and all the randomly generated polynomials $f_{i,update}(x)$. It can be seen from the following proof that S_i' is indeed the partial secret share generated from $f_{new}(x)$.

Proof:

$$f_{new}(x) \equiv f(x) + \sum_{j=1}^{m} f_{j,update}(x) = a_0 + (a_1 + \sum_{j=1}^{m} b_{j,1})x + ... + (a_{k-1} + \sum_{j=1}^{m} b_{j,k-1})x^{k-1} \bmod N,$$

$$S_i' = S_i + \sum_{j=1}^{m} S_{i,j} = f(i) + \sum_{j=1}^{m} f_{j,update}(i) = f_{new}(i).$$

Figure 2 illustrates the share update scheme.

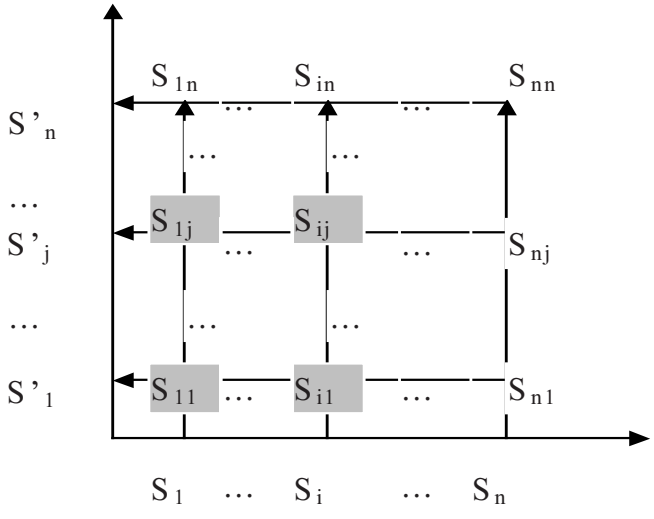

Fig. 2. The proactive share update scheme.

4 Discussion

4.1 Reliability and Intrusion Tolerance

Reliability and intrusion tolerance of the proposed system rely on the three factors: strong cryptographic algorithms, replication and caching mechanisms of the P2P network, and a (*k, m*) threshold secret sharing scheme.

Strong encryption algorithms such as AES and RSA provide robust content protection and ensure that only the authorized users can access the content.

Many P2P networks provide desirable properties such as replication and caching. Copies of the content are well distributed in the system. If a node is unavailable or a file is unavailable in some nodes, there are still many other nodes or file copies in the system which guarantees fault tolerance of the system.

Using a (*k, m*) threshold secret sharing scheme, the secret is divided into many partial secrets and distributed to many LAs. Attackers have to compromise *k* or more LAs to break the system. When *k* is large enough, the proposed system is really intrusion-tolerant.

4.2 Security of the DRM Proposed System

There are many communication sessions between nodes and LAs and among LAs. It is necessary to protect the security of these communication channels. The Secure Sockets Layer (SSL) [31] is used to ensure the communication security. LAs carry certificates to protect them from being impersonated by attackers. The partial result verification mechanism in the step 3 of formal-license generation described in Section 3.4 also thwarts invalid responses. The DRM module running at the consumer's machine also plays a critical role for the security of the system. Its security is exactly the same as the conventional DRM system so the technologies used in the conventional DRM system can also be used in our system.

5 Conclusion

In this paper, we have proposed the Public License Infrastructure (PLI) and the License Authorities (LAs) which are used to build a distributed DRM license service system. Based on PLI and LAs, a novel distributed DRM system has then been proposed for Peer-to-Peer networks. The proposed system uses a (*k, m*) threshold secret sharing and proactive shares update. The threshold secret sharing and the distributed PLI make the proposed system intrusion-tolerant, fault-tolerant, flexible, scalable, reliable, and highly available. Complex and centralized license servers in a conventional DRM system are no longer needed. The license service in the proposed system is provided collectively by group of redundant LAs.

References

1. R. Iannella, "Digital Rights Management (DRM) Architectures," *D-Lib Magazine*, vol. 7, no. 6, June 2001.
2. A. M. Eskicioglu, J. Town, and E. J. Delp, "Security of Digital Entertainment Content from Creation to Consumption," *Signal Processing: Image Communication, Special Issue on Image Security*, vol. 18, no. 4, April 2003, pp. 237–262.
3. Adobe EBooks, available at http://www.adobe.com/epaper/ebooks.
4. Intertrust, available at http://www.intertrust.com.
5. Microsoft Windows Media Digital Rights Management, available at http://www.microsoft.com/windows/windowsmedia/drm.aspx.
6. Napster website, available at http://www.napster.com.
7. Gnutella website, available at http://www.gnutella.com.
8. Architecture of Windows Media Rights Manager, available at http://www.microsoft.com/windows/windowsmedia/wm7/drm/architecture.aspx.
9. Napster, available at http://www.napster.com.

10. Gnutella website, http://www.gnutella.com.
11. JXTA, available at http://www.jxta.org.
12. I. Clarke, B. Wiley, O. Sanberg, and T. Hong, "Freenet: A Distributed Anonymous Information Storage and Retrieval System," *Proc. Int. Workshop on Design Issues in Anonymity and Unobservability*, Springer Verlag, LNCS 2009, 2001.
13. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan, "Chord A Scalable Peer-to-peer Lookup Service for Internet Applications," *Proc. ACM SIGCOMM'01*, San Diego, California, USA, 2001.
14. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," *Proc. ACM SIGCOMM'01*, San Diego, California, USA, 2001
15. A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems," *IFIP/ACM Int. Conf. Distributed Systems Platforms (Middleware)*, 2001.
16. B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph, "Tapestry: An Infrastructure for Fault-tolerant Wide-Area Location and Routing," *Technical Report No. UCB/CSD-01-1141*, Univ. of California, Berkeley.
17. A. Shamir, "How to Share a Secret," *Communications of ACM*, vol. 24, no. 11, 1979, pp. 612–613.
18. V. Shoup, "Practical Threshold Signatures," *Proc. EUROCRPT'00*, Springer Verlag, LNCS 1807, 2000, pp. 207–220.
19. A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive Secret Sharing, or: How to Cope with Perpetual Leakage," *Proc. CRYPTO'95,* Springer Verlag, LNCS 963, 1995, pp. 339–352.
20. C. Cachin, K. Kursawe, A. Lysyanskaya, and R. Strobl, "Asynchronous Verifiable Secret Sharing and Proactive Cryptosystems," *Proc. 9th ACM Computer & Comm. Security*, Washington D.C., 2002.
21. T. Wu, M. Malkin, and D. Boneh, "Building Intrusion Tolerant Applications," *Proc. 8th USENIX Security Symp*., pp. 79–91, 1999.
22. L. Zhou and Z. J. Haas, "Securing Ad Hoc Networks", *IEEE Networks*, vol. 13, no. 6, 1999, pp. 24–30.
23. H. Luo and S. Lu, "Ubiquitous and Robust Authentication Services for Ad Hoc Wireless Networks", *Technical Report 200030*, Dept. of Computer Science, Univ. Of California, Los Angeles, 2000.
24. J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks," *IEEE 9th Int. Conf. Network Protocols*, Nov. 2001, pp. 11–14.
25. H. Luo, P. Zerfos, J. Kong, S. Lu, and L. Zhang, "Self-Securing Ad Hoc Wireless Networks", *IEEE 7th Int. Symp. Computers & Comm*. July 2002, pp. 567–574.
26. The Advanced Encryption Standard (AES), *FIPS-197*, also available from http://csrc.nist.gov/CryptoToolkit/aes/
27. eXtensible Rights Markup Language (XrML), available at http://www.xrml.org/
28. Extensible Access Control Markup Language (XACML), available at http://xml.coverpages.org/xacml.html.
29. Open Digital Rights Language (ODRL), available at: http://www.odrl.net.
30. B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd ed., John Wiley & Sons, Inc. 1996.
31. SSL 3.0 Specification, available at http://wp.netscape.com/eng/ssl3/.