On the Global Content PMI: Improved Copy-Protected Internet Content Distribution

Tadayoshi Kohno and Mark McGovern

Software Security Group, Cigital
{kohno,markmc}@cigital.com

Abstract. This article addresses a problem with copy-protecting a large collection of electronic content. The notion and severity of a *generic attack* are raised in the context of Adams and Zuccherato's Privilege Management Infrastructure. A solution is then proposed that reduces a content distributor's risk of piracy.

Key words: content distribution, copy-protection, PMI, risk management.

1 Introduction

The Internet is changing the way companies do business. News agencies such as the New York Times and CNN publish volumes of online articles daily. The ACM Digital Library sells online access to thousands of journal articles and conference proceedings; and the U.S. Patent Office sells copies of patents online. Sony and Seagram have recently announced that they will distribute music from their websites on a subscription basis. And software companies are beginning to rent and sell software over the Internet.

The Internet is clearly becoming one of the preferred methods for all forms of electronic content distribution: documents, music, images, videos, and software. Unfortunately, without some form of copy-protection, content distributed online could be bought once and then illegally redistributed *ad infinitum*. Under certain business models, such unrestricted redistribution could pose a threat to the financial stability of companies that depend on Internet content sales for their survival.

The protection against and prevention of illegal copying and redistribution of electronic content is called copy-protection. In general, copy-protection schemes are not perfect. That is, in general, no copy-protection mechanism will prevent a determined attacker with unlimited resources from making and distributing illegal copies of copy-protected data. A fundamental goal (and the goal we wish to discuss in this article) is to design copy-protection schemes that *minimize* the illegal copying and redistribution of copy-protected content.

The copy-protection problem is compounded when one tries to protect a large collection of titles using a single copy-protection technique. For example, the Privilege Management Infrastructure (PMI) described in [AZ00] is susceptible to a *generic attack* — an attack that, once found, could be used to un-copy-protect *everything* distributed through that PMI. The presence of generic attacks on copy-protection mechanisms can be devastating. Consider, for example, a company that rents ten thousand different software packages online. If a cracker can figure out a generic attack against the copy-protection scheme used, then the cracker could, with very little effort, automate the removal of the software protection mechanism from all ten thousand packages and distribute those packages (or the automated tools that performs the generic attack) from his or her pirate website.

This article considers a strategy one could use when copy-protecting a large collection of electronic content. That is, this article considers techniques that are resistant to the generic attack described above. The proposed solution is one of risk management. It is a heuristic solution that involves increasing an attacker's work-factor while maintaining an acceptable cost for the content distributor.

The remainder of this article is organized as follows. This article opens with a discussion of terminology (Section 2) and a summary of the Adams and Zuccherato PMI proposal [AZ00] (Section 3). The notion of a *generic attack* is then further developed in Section 4.

Section 5 introduces a PMI variant for protecting dynamic content (such as software) and Section 6 shows how to modify the dynamic content PMI variant so that it is less susceptible to a generic attack (with certain caveats that will be discussed later). The discussion in Section 6 centers around the notion of risk management. The article closes in Section 7 with a summary of results.

2 Terminology

The term *content* refers to any form of digital information that has *value* (to some here unspecified entity). Typical forms of content include electronic documents, music, images, videos, and executable code.

A set of digital information is *dynamic content* if that content executes and if it is the execution of that content that has *value*. More generally, *dynamic content* is content whose output or appearance varies depending on input. A CAD program or a computer game are examples of *dynamic content*. A set of digital information is *static content* if that information has *value* when some (typically external) application executes on it and if each execution produces the same output. Examples of *static content* include images, music, and videos. For dynamic content, one tries to copy-protect the functionality of that content; for static content, one tries to copy-protect the data itself.

The term *title* refers to a specific piece of electronic content. The game "game.exe" (and associated data files), the image "picture.jpg," and the song "music.mp3" are all *titles*. Note that any given title may contain both *static* and *dynamic* portions.

The term *copy-protection* refers to any technique, protocol, or scheme designed to protect electronic content from illegal copying and redistribution. Any digital content that has had a copy-protection technique applied to it is considered *copy-protected*.

Copy-protection techniques may be technical in nature (e.g., use proprietary hardware or cryptography to prevent unauthorized copying), non-technical in nature (e.g., penalize violators with heavy fines and jail time), or both.

3 Privilege Management Infrastructure (PMI)

In [AZ00] Adams and Zuccherato propose a *Privilege Management Infrastructure* (PMI) designed to prevent attackers from illegally copying and redistributing protected electronic content. The technique proposed in [AZ00] is similar to an approach mentioned (though later discounted) in [DLN96, 491].

The PMI for Internet content distribution works as follows. Suppose a company wishes to sell PDF documents online. That company wants users that purchase those documents to be able to view them but also wants users that do not purchase those documents to not be able to view them. That is, if a user purchases a document and then gives that document to a friend, that friend should not be able to view that document.

PMIs are structured after Public Key Infrastructures (PKIs) and consist of a root *attribute authority*. See Figures 1 and 2. The attribute authority signs *attribute certificates*. Attribute certificates bind customer identities with certain content access rights or privileges. In the tradition of [WC87], a user's access privilege list for a given title is called his or her *right-to-execute* (RTE). The RTE may specify unlimited usage (such as in the purchase of a title) or limited-time usage (such as in the rental of a title). The attribute certificate may also contain additional information about the purchased title.

The PMI also consists of a PMI-enabled PDF viewer. Embedded in the PDF viewer is the public key of the root attribute authority. The PDF viewer uses this public key to verify the authenticity of a user's attribute certificate. The PDF viewer also has a copy of a root certificate authority's public key (to verify the identity of the user) and an embedded master symmetric encryption key.

Attribute Authority	**PDF Viewer**	**User**
AA private key	AA public key	
Master symmetric key	Master symmetric key	
	Customer symmetric key	Customer symmetric key
Title symmetric key	*Title symmetric key*	

Fig. 1. PMI-related keys known to the Attribute Authority (content distributor), the PDF Viewer, and the user. A purchased title is encrypted under its title symmetric key. The italicized keys are not stored in the PDF Viewer but are known to the Viewer when it decrypts a title.

Attribute Certificate Contents
User identity information (PKI certificate)
User right-to-execute (RTE)
Doubly-encrypted title symmetric key

Fig. 2. Attribute certificate contents. The title symmetric key is encrypted first under the master symmetric key and then under the user symmetric key.

The protocol for purchasing a PDF document is as follows. The purchaser and the content provider first establish a private, authenticated communications channel. The purchaser then purchases a title (using some standard e-commerce system) and provides the content provider with a customer symmetric key. The content provider encrypts the title with the title symmetric key and then encrypts the title symmetric key first with the master symmetric key and then with the customer symmetric key.[1] The content provider (as an attribute authority) creates an attribute certificate for the customer containing the customer's identity, the doubly-encrypted title symmetric key, and the customer's RTE.

The content provider then sends the user the encrypted title and the attribute certificate. The user authenticates with the PDF viewer and presents the PDF viewer with the attribute certificate and customer symmetric key. After verifying the user's identity, the signature on the attribute certificate, and the privileges specified in the RTE, the PDF viewer decrypts and displays the purchased PDF document.

As with PKIs, PMIs are designed to allow delegation. That is, the root attribute authority can delegate certain rights to other companies or organizations. In the PDF example above, the creator of the PDF viewer would be the root attribute authority and could delegate attribute certificate creation rights to various online magazine publishers.

3.1 PMI Observations

Because later sections of this article build on the PMI, it is important to first consider some of the PMI's limitations and features:

PMIs Versus PKIs. Although PMIs are modeled after PKIs, it is important to note that the trust relationship in PMIs is fundamentally different than the trust relationship in PKIs. In a PKI, when a user or application fails to verify a certificate authorities signature on a certificate, it is usually the user that suffers. In the global content PMI, however, when the PDF viewer fails to verify an attribute authorities signature on a certificate, it is the content provider that suffers. This means that if the PMI PDF viewer is under a user's control and if

[1] The encryption of the title with the title symmetric key and the encryption of the title symmetric key with the master symmetric key could both be performed in a precomputation phase.

the user forces the PDF viewer to ignore the signature on attribute certificates, the user could trick the PDF viewer into displaying documents he or she should not be allowed to view.

Execution Environment. A more general observation is that an attacker with control over the execution environment of a copy-protection scheme will, with enough effort, be able to circumvent that scheme. This observation serves as the basis for our discussions beginning in Section 4 as well as for several secure coprocessor-based copy-protection schemes.

Malicious Distributors. The PMI is vulnerable to a protocol-level attack. In particular, allowing delegation opens the PMI to attacks from malicious distributors. Consider, for example, a malicious distributor of PDF documents. Because the PDF viewer has *one* embedded master symmetric key (i.e., the embedded key does not vary depending on the distributor), a malicious distributor could create valid attribute certificates for a competitor's documents. Although such a distributor might quickly be caught, the potential for "illegitimate" attribute certificates may be a problem in some scenarios.

Identities and Anonymity. Attribute certificates bind user identities (typically PKI certificates) with access rights. A user must have knowledge of the associated private key in order to authenticate with the PMI-enabled PDF viewer. The PMI therefore enforces copy-protection through the "threat of discovery." In particular, one way for a user to illegally distribute protected PDF documents is to distribute his or her attribute certificates along with his or her PKI private key. However, in an ideal world (where certificate authorities validate users' identities before issuing certificates), users will be ill-advised to distribute their identities and private keys.

The use of identities to enforce copy-protection, however, makes anonymity difficult. The PMI may therefore be unsuitable for distributing fringe content or other forms of content with which users may not want their identities associated.

Doubly-Encrypted Content Key. Encrypting the content symmetric keys first by the PDF viewer's master symmetric key and then by the customer-chosen symmetric key does not appear to significantly increase the security of the PMI against theft of content by legitimate users. In particular, the customer symmetric key is superficial; because a user (or attacker) chooses the customer symmetric key, that user could easily strip off the outer encryption of the content symmetric key. The double encryption does, however, appear to aid in the protection of titles against theft by third parties that have learned the master key but do not know any customer symmetric keys.

4 The Generic Attack

As pointed out in the introduction, *generic attacks* on copy-protection schemes can be devastating. A generic attack is an attack on a copy-protection mechanism that, once discovered, can circumvent the copy-protection of *any* title protected by that copy-protection scheme. Consider the PDF PMI described in Section 3. A potential *generic attack* on the PDF PMI might simply consist of an attacker reverse engineering the legitimate PMI-enabled PDF viewer in order to extract the master key. The attacker could then write a PDF PMI extraction program that, given a protected PDF document and a legitimate attribute certificate, decrypts and saves an unprotected version of the PDF document.

If the cracker posts this generic crack to some website (e.g., [Roo00]), then *anyone* (including normal, non-cracker users) could unprotect and redistribute *any* title purchased through the PDF PMI. Although [AZ00] observes that a sophisticated user could circumvent the PMI copy-protection scheme, the assumption in [AZ00] is that the sophisticated user would do so only for his or her own purposes; [AZ00] does not address the presence and significance of a *generic attack*.

5 A Dynamic Content PMI

We now focus on copy-protecting dynamic content and, in particular, software. In this section we describe a dynamic content PMI in which protected titles themselves authenticate users and check for appropriate attribute certificates (in contrast to the PMI-enabled PDF viewer of Section 3). As with the original PMI in Section 3, part of the security of the dynamic content PMI rests in an attacker's inability to reverse engineer and tamper with executable code.

In Section 6 we discuss how to convert the dynamic content PMI into an approach resistant to the generic attack.

Preliminaries. Let P refer to a dynamic content publisher and attribute authority. Let A refer to a legitimate user that wishes to purchase a title and let T refer to the software title the user wishes to purchase. Let L refer to an executable module that wraps and decrypts T. Let K_T refer to the the title's symmetric key, let K_A refer to A's symmetric key, and let K_L refer to the key embedded in the loader L.

Let I_A represent A's identity with respect to some PKI and let $R_{A,T}$ represent A's access privileges (RTE) with respect to title T. In addition to decrypting and running the content T, the loader L is responsible for authenticating the user and checking the user's attribute certificate for the appropriate RTE.

The Purchase Protocol. The dynamic content PMI distribution algorithm proceeds as follows. In the precomputation stage, P selects a randomly distributed key K_T and then encrypts the content T using the title key K_T. The encrypted title is then bundled with a loader L to create an executable T'.

After performing the necessary e-commerce transactions to purchase a title T, A and P establish a private, mutually authenticated channel. A then sends P his or her identity information I_A and symmetric key K_A.

P doubly-encrypts the title key K_T first with the loader key K_L and then with the user key K_A. P then creates and signs an attribute certificate X composed of I_A, $R_{A,T}$, and the doubly-encrypted key K_T. P sends this attribute certificate to A.

Playing the Purchased Title. To play the purchased title, the user runs T' with input X and K_A. After T' verifies P's signature on X, the user authenticates with T' using his or her private key. T' then decrypts and runs the original title T with the permissions specified in the RTE $R_{A,T}$.

Incorporating the Loader L with the Title T. The above description assumes that the software distributor P retrofits titles T with loaders L in order to produce protected titles T'. Such retrofitting is primarily applicable when P is a third party distributor not involved with the development of T. A better solution, however, would be to intersperse access checks and other protection mechanisms throughout T.

Attacks on Self-Decrypting Executables. As with any cryptographic system, one should not confuse privacy (and encryption) with authenticity. A user of the dynamic content PMI should therefore be cautioned that unless he or she receives a "protected" software title T' through a mutually authenticated channel (as described in *The Purchase Protocol* above), the executable T' may contain trojan, virus, or other malicious code and should not be trusted. This problem is common to all self-decrypting executables.

Platform Dependence. One of the advantages of the original global content PMI [AZ00] is that it was designed to allow customers to access purchased content on any appropriate device. It is therefore prudent to note that, because both the dynamic content itself and the loader may be platform dependent, the dynamic content PMI may be platform dependent. This observations remains true even when the dynamic content PMI is used to protect static content (Section 5.1) unless the loader is written in a platform independent manner.

5.1 The Dynamic Content PMI with Static Content

This section shows how to adapt the dynamic content PMI for use with static content. There are several caveats to this approach. For example, this approach will increase the bandwidth requirements for static content distribution. Changing static content into executable content could also create another channel for the distribution of viral or malicious code. Additional caveats will be discussed in Section 6.3.

The general technique is to bundle the static content with a viewer V in much the same way that an executable T is bundled with a loader L in the above dynamic content protocol. For example, to copy-protect digital images, a content provider could wrap each image in a Java applet that checks for an appropriate attribute certificate before decrypting and rendering a picture. This technique is very similar to a technique proposed by Petitcolas, Anderson, and Kuhn to defeat web-crawling watermark detectors [PAK99, 1071]. Although there are obvious flaws with this approach, static content distributed this way is no more susceptible to illegal redistribution than unprotected static content.

Section 6.3 raises additional concerns with using the dynamic content PMI to protect static content (and presents additional motivation for distinguishing between the protection of static content and dynamic content).

6 Risk Management and Per-Title Copy-Protection

The copy-protection problem in an insecure environment exemplifies the fact that there are seldom absolutes in computer security: The question is not whether the dynamic content PMI in Section 5 is secure — the question is *how* secure the dynamic content PMI is and how much *work* must an attacker exert to break it.

While one could certainly modify the dynamic content PMI for use with secure coprocessors (such that only trusted coprocessors could decrypt and execute critical portions of the protected title), we shall restrict ourselves to software-only copy-protection.[2]

6.1 Risk Management

As with the standard PMI (Section 3), the dynamic content PMI in Section 5 is susceptible to a generic attack. To paraphrase Section 4, a generic attack against a copy-protection scheme is extremely devastating because an attacker could use the attack to break *any* title protected by the copy-protection scheme. For example, suppose an attacker creates a generic attack tool that, given a protected title T' and an attribute certificate X, creates an executable title T'' functionally equivalent to the original, un-protected title T. The attacker could then use the generic attack tool to un-protect any title distributed through the dynamic content PMI.

Obviously, the content distributor would prefer for *none* of the titles it distributes to be attacked. However, as noted above, the question is not whether an attacker could circumvent the copy-protection mechanism, but how much *work* an attacker would have to exert in order to do so. In order to justify that work, the attack must be highly profitable for the attacker. This leads to the

[2] As secure coprocessors become more prevalent, a secure coprocessor PMI may become a more viable solution (in addition to other secure coprocessor-based schemes; e.g., [PSS82,WC87,HP87,YT95,GO96]). However, if the coprocessors used in a copy protection scheme are vulnerable to tampering attacks or side-channel analysis, then the secure coprocessors become insecure processors and the strategy discussed in this section can be used to increase the security of the protection mechanism.

notion of a *work-factor*, or the ratio of the effort an attacker must exert in relation to the resulting profit or yield. The higher the work-factor, the better the copy-protection mechanism.

For example, a professional pirate might be justified in spending a solid month to create a generic attack that could be used to un-copy-protect a thousand titles valued at a hundred U.S. dollars each. The work-factor in this scenario is very low. The same pirate would be hard-pressed to justify spending the same amount of time to break a copy-protection mechanism that is only used with one (or perhaps even a few) similarly priced titles because the work-factor would be much greater.

To compliment the desire to maximize an attacker's work-factor, the proposed solution must be efficient for the content distributor. This means that the content distributor should not have to exert a large amount of extra work in order to increase an attacker's work-factor. The appropriate balance between the advantage gained by increasing an attacker's work-factor with the amount of extra work a content distributor must perform will depend on the content distributor's business model and the value of the protected content.

The solution proposed in this paper is one of *per-title copy-protection* — protecting each title in a slightly different way. This could have three results: (1) the attacker would have to exert much more time and effort to break all the titles distributed by the content provider, (2) the attacker would become discouraged during the process of breaking individual titles and give up, or (3) the attacker would realize the futility in attacking the system. Obviously (2) and (3) are the preferred results. But even if a protection mechanism only succeeds in (1), that protection mechanism is still useful — it successfully *increased* the copy-protection afforded each title.

6.2 Per-Title Copy-Protection

Before proposing a method for per-title copy-protection, let us consider the ways an attacker might break the copy-protection of a dynamic content PMI-protected title T'. The attacker could exhaustively search the symmetric key K_T or K_L, the attacker could find an attack against the algorithm used to encrypt the title, or the attacker could obtain the content distributor's attribute authority private key. Most likely, however, the attacker would break the copy-protection mechanism through reverse engineering T'. For example, the attacker could defeat the copy-protection mechanism by changing T' so that it no longer attempts to verify the attribute authority's signature on a user's attribute certificate. An attacker could also defeat the protection mechanism by reverse engineering T' in order to obtain K_L.

The point of the above paragraph is not to present a complete taxonomy of attacks against protected titles, but rather to illustrate that most practical attacks will involve the attacker stepping through, understanding, and/or modifying the execution of T'.

The solution proposed here consists of randomized, per-title obfuscation and software tamper resistance [CTL97,MMO97]. According to [CTL97], "code obfuscation is currently the most viable method for preventing reverse engineering." By applying potent and highly resilient obfuscation techniques (see [CTL97]) to each title, the content provider would force an attacker to exert work when breaking each title. This results in an increase in the attacker's work-factor and, consequently, an increase in the security of the copy-protection scheme. A similar approach can be found in [MC98].

In addition to obfuscation, a content distributor could employ other per-title access checks or protection mechanisms. Randomized code obfuscation has an advantage over these additional protection mechanism because code obfuscation is automateable and therefore efficient for content distributors to apply. If the value of the protected title is high, however, the content distributor may be justified in implementing additional, title-specific access checks throughout different components of the title.

Unfortunately, the resulting per-title scheme may still be vulnerable to attacks on a per-title basis. Furthermore, because of potential commonality between protected titles (especially with respect to the transition between the PMI access checks and the execution of the title itself), the per-title protection mechanism above does not preclude the existence of more sophisticated generic attacks. However, if the obfuscation techniques used are highly resilient, creating such a generic attack may be exceedingly difficult and would be of independent interest.

6.3 The Dynamic Content PMI with Static Content (Revisited)

Although the dynamic content PMI is, by definition, designed to protect dynamic content (such as software), Section 5.1 showed that the dynamic content PMI could also be used to protect static content (such as documents, images, and videos). There are, however, some fundamental differences between static and dynamic content that make the per-title dynamic content PMI more suitable for dynamic content than static content.

The biggest problems with using the dynamic content PMI (and similar) techniques to protect static content is that the PMI protection mechanism has no control over what happens to static content after the content is displayed to the end user. This leads to an exploitable disassociation between the protection mechanism (the loader) and the protected content — an attacker might attack the dynamic content PMI (for static content) by stealing the content after the loader verifies the user's attribute certificate and presents the title (rather than by attacking the loader itself). This disassociation remains even if the protection mechanism checks for permission periodically *throughout* the play or rendering of the static content.

To further develop this notion, first observe that because static content must eventually be displayed to the end user in order to have value, an attacker able to intercept that display channel will be able to copy that data (at a potential loss in quality). Second and more importantly because static content does not vary between views, an attacker able to steal a copy of *one view* of a static title will have obtained all the value of that title. This is compared to stealing a "trace" of a single execution of some dynamic content such as a game — after the game is played, the trace has very little value.

The proposed dynamic content solution attempts to "glue" together (on a per-title basis) the functionality (value) of dynamic content with the protection mechanism. Additional security (perhaps at additional developer expense) could be obtained by permeating a variety of protection mechanisms throughout each component (and hence the execution) of a title.

7 Conclusions

This article addresses a problem with copy-protecting a collection of electronic content. Software-based copy-protection of electronic content in an attacker-controlled environment is adequate at best. An attacker can capture static content (e.g., images, music, and videos) as the content passes between some decoding device and an end user. And an attacker can disassemble dynamic content (e.g., software) and remove the content's copy-protection mechanism. Assuming that all software-based copy-protection mechanism are breakable given enough effort, this article presents a strategy to reduce a content provider's risk of content piracy.

This article begins with a discussion of Adams and Zuccherato's Privilege Management Infrastructure (PMI) [AZ00] (Section 3). Several attacks against the PMI are discussed and, in particular, Section 4 presents a *generic attack* against the PMI. A *generic attack* is an attack against a copy-protection system that, once found, can be used to break the copy-protection of *all* content protected through that system.

Sections 5 and 6 show how to modify the PMI so that it is less vulnerable to a generic attack. Although developed in the context of Adams and Zuccherato's PMI, the general principle of *per-title copy-protection* presented in Section 6 can be used in conjunction with other copy-protection schemes.

Although the solution presented here may be disheartening to those who prefer provably secure protocols, this article argues that because content copy-protection in attacker-controlled environments (e.g., without secure hardware) may be an unsolvable problem, any cost-effective (e.g., efficient to apply; not inordinately complex) increase in security is advantageous. This is analogous to the state of the art in watermarking (as described in [CT98]) where the philosophy is to provide as many layers of protection as possible in order to prevent all but the most dedicated attacker.

Acknowledgments

The authors thank Tim Hollebeek, Gary McGraw, Win Treese, Robert Zuccherato, and the anonymous referees for invaluable comments and suggestions.

References

[AZ00] C. Adams and R. Zuccherato. A global PMI for electronic content distribution. In *Seventh Annual Workshop on Selected Areas in Cryptography*. Preproceedings, 2000. Springer-Verlag, to appear.

[CT98] C. Collberg and C. Thomborson. On the limits of software watermarking. Technical Report 164, Department of Computer Science, University of Auckland, Auckland, New Zealand, 1998.

[CTL97] C. Collberg, C. Thomborson, and D. Low. A taxonomy of obfuscating transformations. Technical Report 148, Department of Computer Science, University of Auckland, Auckland, New Zealand, 1997.

[DLN96] C. Dwork, J. Lotspiech, and M. Naor. Digital signets: Self-enforcing protection of digital information (preliminary version). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, 1996.

[GO96] O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious RAMs. *Journal of the ACM*, 43(3), 1996.

[HP87] A. Herzberg and S. S. Pinter. Public protection of software. *ACM Transactions on Computer Systems*, 5(4), 1987.

[MC98] S. A. Moskowitz and M. Cooperman. Method for stega-cipher protection of computer code. US Patent 5745569, April 1998.

[MMO97] M. Mambo, T. Murayama, and E. Okamoto. A tentative approach to constructing tamper-resistant software. In *Proceedings of the Workshop on New Security Paradigms Workshop*, 1997.

[PAK99] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn. Information hiding: A survey. *Proceedings of the IEEE, Special Issue on Protection of Multimedia Content*, 87(7), 1999.

[PSS82] G. B. Purdy, G. J. Simmons, and J. A. Studier. A software protection scheme. In *1982 Symposium on Security and Privacy*, 1982.

[Roo00] Rootshell. http://rootshell.com/, 2000.

[WC87] S. R. White and L. Comerford. ABYSS: A trusted architecture for software protection. In *1987 IEEE Symposium on Security and Privacy*, 1987.

[YT95] B. Yee and J. D. Tygar. Secure coprocessors in electronic commerce applications. In *First Usenix Workshop on Electronic Commerce*, 1995.