Digital Rights Protection [*]

Mikhail J. Atallah   Sunil Prabhakar   Keith B. Frikken   Radu Sion
Department of Computer Science,
Center for Education and Research in Information Assurance and Security (CERIAS)
Purdue University
Email: {mja, sunil, kbf, sion}@cs.purdue.edu

Abstract

Digital Rights Protection (DRP) is the broad class of technological, legal, and other regulatory means used to protect the rights of the owners of digital content, while simultaneously protecting the usage rights and the privacy of the users. This article briefly discusses the technological aspect of the issue.

1 Introduction

The ability to perfectly (and cheaply) replicate digital objects (software, data, etc.) has been one of the greatest advantages of the digital format over analog. While this property enables mass reproduction at low costs, it proves to be a double-edged sword since it also allows perfect reproduction for purposes of piracy. Piracy of digital works such as music and software programs is already commonplace today and results in significant losses for the owners of the digital content due to their inability to collect payment for the illegal use of pirated copies. In addition to the ease of copying digital objects, the extensive use of the Internet as a distribution medium allows widespread sharing of illegal copies. The goal of Digital Rights Protection (DRP) is to protect the Intellectual Property rights of owners of digital content. In addition to protecting the rights of owners of digital content DRP also aims to protect the usage rights and privacy of the (legitimate) consumers of digital content. The ability to share information in digital format over the Internet is highly attractive from the viewpoint of ease, speed and cost. However, the fear of piracy prevents many individuals and corporations from fully embracing this alternative. Sound DRP techniques are essential in removing this fear and encouraging more effective use of the Internet. But it is all too easy for DRP technologies to encroach on the user's rights, such as privacy (if the technologies allow tracking of the user's access to the data), or "fair use" (such as making a private copy for convenience, e.g., for use on another platform owned by the user).

DRP is challenging because it is not sufficient to defeat an average attacker, or even most attackers – rather one has to protect against the best attackers. This is because even a single compromise of the protection mechanism can lead to an unprotected copy of the digital object which can then be widely distributed. Moreover, the ability to easily share the instructions for compromising the protections over the Internet renders each user

[*]

potentially as powerful as the most skilled attacker that breaks the protection system and shares the methodology (often through point-and-click automated scripts).

A distinction between DRP and access control can be made based upon the location of the protected object: In the case of access control, the digital object resides in the content owner's (server's) space; In the case of DRP, the object resides in the client's space. Since objects residing in the user's space can be freely accessed by the user, DRP faces unique challenges in preventing unauthorized access to and use of the data. DRP is also different from the related field of Privacy Preserving data access/sharing and Secure Multi-Party Computations. The goal of privacy preserving data access is to ensure that only desired aspects of the data are shared while hiding other aspects. For example, in privacy preserving data mining across multiple domains that do not fully trust each other, the goal is to discover interesting patterns over the union of the data without revealing the data owned by another participant to any of the participants.

2 Current Approaches

Current approaches to DRP can be divided into two broad categories based upon the nature of the protected digital object: Software or Data. Software DRP deals with ensuring that programs are not illegally replicated and disseminated. Data DRP deals with rights protection for digital objects such as images, video, and databases. Enforcing DRP for data can often be achieved through controls placed on the software. For example, videos can be distributed in a format recognized only by special software. Ensuring that this software is used only for authorized playing of these videos through software DRP methods results in protection for the media.

We now briefly discuss some of the mechanisms for protecting digital rights of copyright owners.

2.1 Watermarking

Digital watermarking aims to protect digital content by enabling provable ownership over content. This is achieved by modifying the digital objects such that identification information is embedded in the object itself. With regards to the resilience of the watermark, there are two categories: *robust* and *fragile*. A robust watermark must be resilient to attack, i.e. it must not be easy to remove the watermark without significantly destroying the object itself and thereby rendering it useless. On the other hand a fragile watermark is easily destroyed by even minor alterations to the data. Each type of watermark serves its own purpose. In addition to proving ownership of content, watermarking can also be employed to trace copyright violations and detecting modification.

An essential property of a watermark is that it modifies the digital object. It is important that the modification have minimal impact (if any) on the use of the digital object. The degree of modifications allowed depends on the nature of the object and its application. For multimedia objects such as images, acceptable change is invariably defined by the inability of the human sensory system to distinguish between the original and watermarked objects. For the case of software, the modification must ensure that the resulting code perform equivalent computation to the original object (at least with respect to the desired functionality).

Attacking a watermarked objects entails further modification of the object such that the watermark is no longer detectable. As with the insertion of the watermark, any attempt at deleting the watermark can only be considered successful if in the process the usability of the object is not destroyed. Attacks techniques include *additive* attacks that insert more data, *subtractive* attacks that throw away a large portion of the data, the insertion of another watermark, and modifications. Modifications to the object are most effective if the attacker determines what parts of the object have been modified to insert the watermark. Consequently, many watermarking schemes take great pains to hide this information through the use of secret keys. If the attacker is unable to precisely identify the location of the watermark then the modifications must necessary be random, which increases the likelihood of undesirable destruction of the value of the object. A good watermarking scheme should therefore try to maximize this likelihood, Digital watermarks are most often used for establishing proof of ownership by inserting a robust watermark into the digital object. Ownership of an object is established through the detection of the watermark by using a secret (possibly the same secret key used in the generation of the watermark). In order to be convincing, the watermark should be such that it would be highly unlikely that the watermark could occur by accident. A good watermarking method should be immune to the "torturing the data" defense. This line of defense claims that the alleged owner of the object has exhaustively searched this digital object using various keys and has thereby "discovered" this watermark. Such claims can be countered by publishing a one-way cryptographic hash of the secret key in a time-stamped manner (such as in a newspaper).

Inserting different watermarks in different copies of the same object can be used to record information about each copy, in addition to information about the owner of the copyright. This additional information can include the identity of the purchaser of that copy. Such watermarks can be used as *fingerprints* in order to trace copyright violations. A pirated copy would bear a watermark that uniquely identifies the initial purchaser of this copy. A down-side to fingerprinting is that it is vulnerable to a special kind of attack – the "diff" attack. In his attempt to remove a watermark, an adversary can be much more effective if he can identify the locations of the watermark. These locations are typically hidden through the use of secret keys during embedding. With the availability of multiple watermarked versions of the same data, a simple comparison easily identifies potential locations of the watermark. These can then be altered yielding the watermark ineffective without having to make major modifications to the object. Although fingerprinting can help identify the original recipient (and thereby the source of the piracy) the culpability of the recipient is a different matter – e.g. one could claim that the object was stolen and pirated by the thief.

The ability of robust watermarks to survive significant modifications can be used to serve other purposes that require tamper-resistance. For example, a watermark that attests to some fact can serve as a credential. The content of the watermark text can be used to embed control information such as restrictions on the use of the data. For example, a media player may only play a file if it contains a watermark with the appropriate permissions embedded in it. On the other hand, the intolerance of a fragile watermark to even minor changes can be used as a means for establishing the integrity of the data. The absence of the watermark is an indication that some unauthorized modification of the file has taken place.

The most common application of digital watermarking has been for the domain multimedia data such as images and video. The watermark is typically embedded in either the original domain of the object, such as by modifying the low order bits of pixel values, or in more sophisticated methods, in the corresponding frequency domain. In frequency domain watermarking, the object is first converted to the frequency domain through transformations such as the Fast Fourier Transform (FFT) or the Discrete Cosine Transform (DCT). The watermark is embedded in this domain through minor modifications, and then the watermarked object is generated through the corresponding inverse transformation back to the original domain. Due to the reliance of these methods on the relative insensitivity of the human sensory system, and the regular structure of media data, these watermarking methods are typically not applicable to other domains such as software and data. Watermarking has also been applied to software, natural language text, and more recently, structured and semi-structured data. Watermarking of software is achieved through modifications to the structure of the code (such as IBM's use of a very specific register loading order) or its behavior. Watermarks that modify behavior may be detectable at run-time only if a secret key is provided.

Watermarking structured data, such as a relational database is an interesting problem that has received little attention thus far. A unique feature of this domain is that the issue of acceptable change is not as clear as in other domains. Unlike the multimedia domain, the insensitivity of the human sensory system is not available since even minor changes in some critical data values can be important. More importantly, acceptability of changes to data values is highly application dependent.

Overall, watermarking is an important tool for Digital Rights Protection since it enables proof of ownership of content, traceability of pirated copies, integrity testing, and authentication.

2.2 Code Obfuscation

In instances where the intellectual property to be protected lies in the algorithms or data structures in a piece of software, it is necessary to ensure that malicious users are not able to reverse engineer the code. Code Obfuscation can be used to achieve this purpose. Code Obfuscation is the process of mangling code and data structures used in the code so that it becomes difficult to understand the functioning of the program. These methods are also valuable in making protection mechanisms in the code harder to defeat or circumvent. For example, a simple implementation of a software check that requires a password or key to work correctly typically contains a conditional jump statement that makes the critical decision of whether or not the password or key is valid. If an attacker is able to discover this statement (for example using code cracking tools such as those discussed below), the check can easily be defeated by changing the jump to an unconditional jump or other similar change. Code obfuscation cannot guarantee that the attacker will not be able to reverse engineer the obfuscated code. However, it can make the task highly non-trivial rendering it difficult for a novice attacker to succeed, and at least significantly slowing down expert adversaries.

In order to be effective (i.e. resilient to attack) and efficient, Code Obfuscation has to meet certain requirements, including: 1) the original functionality of the software must remain unchanged; 2) the modifications should not be easy to remove using automatic tools; 3) the resulting performance degradation should be acceptable; and 4) the modifications should not be removed by a compiler during compilation. Obfuscation techniques fall into one of the following categories:

1. *Data* obfuscation reorganizes data structures and the code that uses them so as to obscure their role. Examples include merging of unrelated data, or taking apart data that are related.

2. *Control* obfuscation aims to hide the true control flow of the program by entwining it with segments of code that have no actual influence on the control structure of the program. The modification must make it difficult to identify these irrelevant portions of the resulting program. Naturally, control obfuscation must survive the compilation process – changes that the compiler is able to discard due to their irrelevance (such as a function that is never called) will not work.

3. *Layout* obfuscation deals with the rewriting and rearranging of the source code of a program to make it harder to read. Examples of layout obfuscation include the removal of comments, variable substitutions, and altering the formatting of statements.

An important tool for obfuscating segments of code is the use of *opaque predicates*. An opaque predicate is a conditional test that will always generate the same answer although it would be very difficult to arrive at this conclusion simply through an examination of the code. Opaque predicates can be inserted in code to give the appearance of being integral parts of the code whose “role” is difficult to discern.

Methods of Attack We briefly mention some of the methods that can be employed to defeat rights protection mechanisms discussed above. An important step in circumventing software protections is to first understand the functioning of the code. Commonly available tools for debugging, decompiling or disassembling tools can be used to trace the control flow during execution of the program, locate interesting events in the code, and generate “cleaner” versions of the code. Many protection related segments of code perform encryption. Performance profilers and pattern matching can be used to identify such pieces of code, which can then be subjected to attack.

2.3 Other Methods

Encryption Wrappers One way to hide code or data is to store it in encrypted format using a secret key. This data is automatically decrypted when it is needed. Since the data is to be decrypted on the user’s computer, an attacker would be able to obtain the decrypted version of the data. Thus the key to this technique is to prevent the user from easily obtaining the decrypted version. This is achieved through techniques such as preventing the use of tools that can dump the decrypted code to disk or debugging the program at run-time. Alternatively, the software decrypts the program in small chunks thereby making it difficult to obtain a snapshot of the entire program at any one time. Thus an attacker must obtain numerous snapshots and integrate them to obtain a copy of the entire encrypted program. A weakness of encryption wrappers is that the decryption key is contained within the encrypted data and thus can be discovered through an analysis of the program.

Hardware Techniques Although software techniques for DRP have been studied and used extensively, it has been established that it is impossible to have completely secure protection using software alone. This makes hardware protection highly desirable if strong guarantees of protection are needed. Unlike software protections, defeating hardware protections is much harder and may require special equipment. In addition, even if a hardware protection mechanism is defeated by a determined and expert attacker, it may not be as easy to share the compromise with others.

Hardware-based DRP methods make use of devices such as trusted hardware processors or peripheral devices (often called Dongles). Typically, dongles are small devices that are connected to the client computer and interact with the protected program. The role played by the dongle can vary from simply establishing the right to use a digital object by their presence, to playing an essential role in the program. The use of dongles is expensive for the producer of the digital content and may be inconvenient for the user since a dongle is needed for each copy of protected software. Trusted processors provide a means for executing only software that has not been corrupted (such as bypassing a critical authorization check) – this includes all types of code run on the machine including the operating system. The extensive use of encryption that goes along with the use of these trusted processors can be prohibitive for a low-end market. In addition, it can lead to limitations on the use of the system for other purposes that are not related to the legitimate protection of digital rights.

2.4 DRP for Databases

The problem of protecting ownership rights over collections of data such as in a database, or XML format has recently begun to draw the attention of researchers. It has become clear that existing DRP techniques cannot directly be ported to this domain. Consider the case of watermarking wherein the notion of acceptable change is highly dependent upon the intended use of the data. An encrypted version of the data with minor changes may be acceptable change for some applications while not for others. The current results on watermarking of relational data are only the first steps towards enabling robust DRP solutions for structured data.

3 Conclusion

The ease of replication and distribution of digital media make it attractive for content owners as a means of cheap, rapid and widespread distribution. However, these very same properties make it difficult to assert ownership rights and collect payment over the same objects. Digital Rights Protection mechanisms therefore play an important role in enabling content owners to prove ownership of digital content and to limit illegal distribution and use of digital content. Both software and hardware techniques have been developed that provide a wide variety of protections. Software only solutions cannot guarantee protection – and may eventually be compromised by a determined and expert adversary. Thus these solutions achieve the purpose of being deterrents and delaying the adversary. The focus of DRP solutions has been software and media objects such as images. The problem of protecting ownership rights over collections of data such as in a database, or XML format has recently begun to draw the attention of researchers. It has become clear that existing DRP techniques cannot directly be ported to this domain. The current results on watermarking of relational data are only the first steps towards enabling robust DRP solutions for structured and data.

References

[1] R. Anderson *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wily, John & Sons, Inc. February 2001, p. 413-452.

[2] R. Anderson "TCPA/Palladium Frequently Asked Questions" *www.cl.cam.ac.uk/ rja14/tcpa-faq.html*.

[3] R. Anderson "Security in Open versus Closed Systems - Dance of Blotzmann, Coase, and Moore" *www.ftp.cl.cam.ac.uk/ftp/users/rja14/toulouse.pdf*.

[4] R. Anderson, M. Kuhn "Tamper Resistance - a Cautionary Note" Proceedings of Second Usenix Workshop on Electronic Commerce, pp. 1–11, November 1996.

[5] R. Anderson, M. Kuhn Low Cost Attacks on Tamper Resistant Devices Proceedings of the 1997 Security Protocols Workshop, Paris, April 7–9, 1997.

[6] W. Arbaugh, D. Farber, and J. Smith "A Secure and Reliable Bootstrap Architecture" Proceedings of the IEEE Symposium on Security and Privacy, 1997.

[7] Mikhail Atallah, Keith Frikken, Carrie Black(*), Susan Overstreet, and Pooja Bhatia "Digital Rights Management" *Practical Handbook of Internet Computing*, Munindar P. Singh (Ed.), CRC Press, 2004.

[8] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan and K. Yang *On the (Im)possibility of Obfuscating Program* Electronic Colloquium on Computational Complexity, Report No. 57, 2001.

[9] T. Budd "Protecting and Managing Electronic Content with a Digital Battery" *Computer*, vol. 34, no. 8, pp. 2-8, Aug 2001.

[10] A. Caldwell, H. Choi, A. Kahng, et. al. "Effective Iterative Techniques for Fingerprinting Design IP" UCLA Computer Science Department, 1999.

[11] J. Camenisch "Efficient Anonymous Fingerprinting with Group Signatures" IBM Research, Zurich Research Laboratory, 2000.

[12] C. Collberg, C. Thomborson, and D. Low *A taxonomy of obfuscating transformations*. Tech. Report # 148, Department of Computer Science, University of Auckland, 1997.

[13] C. Collberg and C. Thomborson *Watermarking, Tamper-Proofing, and Obfuscation Tools for Software Protection*. IEEE Transactions on Software Engineering. Vol. 28 No. 8 pages 735-746, 2002.

[14] M. Jakobsson and M. Reiter "Discouraging Software Piracy Using Software Aging" Digital Rights Management Workshop 2001: 1-12.

[15] A. Kahng, J. Lach, W. Mangione-Smith, S. Mantik, I. Markov "Watermarking Techniques for Intellectual Property Protection" UCLA Computer Science Department, 1998.

[16] R.J. Lipton, S. Rajagopalan, D.N. Serpanos "Spy: a method to secure clients for network services" IEEE Distributed Computing Systems Workshops 2002:23-28.

[17] T. Maude and D. Maude "Hardware Protection Against Software Piracy" *Comm. of ACM*, vol. 27, no. 9, pp. 950-959, Sep 1984.

[18] B. Pfitzmann and A. Sadeghi "Coin-Based Anonymous Fingerprinting" Universitaet des Saarlandes, Fachbereich Informatik. Saarbrcken, Germany. 1999.

[19] B. Pfitzmann and M. Schunter "Asymmetric Fingerprinting" Universitaet Hildesheim, Institut fr Informatik. Hildesheim, Germany. 1996.

[20] G. Qu and M. Potkonjak "Fingerprinting Intellectual Property Using Constraint-Addition" Computer Science Department, University of California Los Angeles, 2002.

[21] G. Qu, J. Wong, and M. Potkonjak. "Optimization-Intensive Watermarking Techniques for Decision Problems"

[22] B. Schneier. *Secrets and Lies: Digital Security in a Networked World.* John Wiley & Sons Inc., 2000.

[23] M. Seadle, J. Deller, and A. Gurijala "Why Watermark? The Copyright Need for an Engineering Solution" JCDL. July 2002.

[24] R. Sion, M. Atallah, S. Prabhakar *Rights Protection for Relational Data* ACM International Conference on Management of Data, SIGMOD 2003,

[25] C. Wang, J. Hill, J. Knight and J. Davidson *Software Tamper Resistance: Obstructing Static Analysis of Programs* PhD Dissertation, University of Virginia.

[26] G. Wroblewski *General Method of Program Code Obfuscation*, PhD Dissertation, Wroclaw University of Technology, Institute of Engineering Cybernetics, 2002