Copyright Protection for Electronic Publishing over Computer Networks

A. K. Choudhury, N. F. Maxemchuk, and S. Paul

AT&T Bell Laboratories
Murray Hill, NJ 07974
{akc,nfm,sanjoy}@research.att.com

*H. G. Schulzrinne**

GMD-Fokus
Berlin, Germany
hgs@fokus.gmd.de

Abstract

The increased availability of computers, printers and high-speed networks could make electronic publishing a reality. One of the major technical and economic challenges faced by electronic publishing is that of preventing individuals from easily copying and illegally distributing electronic documents. In this paper, we explore the use of cryptographic protocols to discourage the distribution of illicit electronic copies. We propose an architecture and two separate schemes for making electronic document distribution secure. The first strategy requires special-purpose firmware in the printers and displays to decrypt encrypted documents. In the second strategy, encrypted documents are decrypted in software in the recipient's computer.

1 Introduction

The increased use of facsimile has made the electronic transfer of paper documents more accepted. Electronic mail, electronic bulletin boards and networks such as the Internet make it possible to distribute electronic information to large groups. Moreover, the proliferation of personal computers and workstations, the excellent quality of desktop printers, and the decreasing cost of storage devices for large volumes of electronic data have made it technologically feasible to display, print, and store documents electronically. All these developments together could make electronic publishing a reality. The electronic distribution of information is faster, less expensive, and requires less effort than making paper copies and transporting them. Other factors that favor electronic information distribution include the ability to use a computer to search for specific information, and the ability to more easily customize what is being distributed to the recipients. Also, recipients may be able to choose the desired method of presentation. Consequently, electronic newspapers, magazines and journals are poised to supplement, and perhaps eventually replace, the current paper distribution networks. In this paper, we shall concern ourselves primarily with the electronic distribution of articles from scientific and trade journals, magazines and newspapers. However, most of the techniques equally apply to multimedia objects such as audio clips or movies. For simplicity, we will use the term *document* throughout.

*This work was performed while the author was at AT&T Bell Laboratories, Murray Hill, New Jersey.

Submitted to
IEEE Network Magazine
June 1994

Electronic document distribution can be roughly divided into two modes: the first stores documents in electronic form, but delivers them through physical media, such as CD ROMs. In the second mode, the one of interest to this paper, the document is stored (or generated on the fly) at a remote site and accessed through a computer network. Network access is particularly appropriate if the data changes frequently, or if the document storage exceeds that available locally [1]. Distribution through networks makes it easy to avoid the traditional update cycles of physical media, such as the periodic publication of journals or the updates of CD ROMs. A journal article, for example, could be "published" as soon as it has completed the editorial process.

For the publisher, the cost of network publication may be significantly lower than having to set up a distribution network, particularly if data access is sparse, i.e., each individual user only accesses a tiny fraction of the total document store. Also, charging can be far more flexible, as discussed later.

Network access to data also allows referencing without having to include the actual reference. For example, an article in a music journal could refer to a recording, owned by a different publisher, that the reader could retrieve, using whatever charging arrangements have been negotiated with the music publisher.

The advantages offered by electronic distribution are also among the primary technical impediments to the acceptance of electronic documents as a replacement for paper versions. One of the major technical and economic challenges faced by electronic publishing is that of preventing individuals from easily copying and illegally distributing electronic documents [2–4]. It is easier for a person who obtains an electronic document to forward it to a large group than it is for a person who receives a paper copy of the same document. In addition, electronic copies are more like the originals than paper copies. When an electronic copy is made, the original owner and the recipient have identical entities. A person with a photocopy of a journal and a person with the original bound journal may have the same information, but it looks and feels different. Illicit copies of electronic documents are likely to result in major losses in revenue.

The ability to custom-tailor copies of a document for each recipient could also be used to identify the original owner of a document. Some recent work has shown that it is possible to mark documents in order to discourage individuals from distributing journals in violation of the copyright laws [5,6]. These references describe how information that identifies the original owner is encoded into the document to make each copy unique. These techniques are complementary to the ones described here.

In this paper, our objective is to explore the use of cryptographic protocols to discourage or prevent the distribution of illicit electronic copies. In Section 2, we discuss various formats in which electronic documents are available. Section 3 lays out the requirements that the protocols for electronic publishing will have to meet in order to become a viable alternative to conventional publishing. In

Submitted to
IEEE Network Magazine
June 1994

Section 4 we present two approaches that address these requirements. We conclude with our view of the viability of electronic publishing.

2 Document Types and Document Value

Unlike traditional media like paper and even video tape, for true electronic documents, the same basic content can be delivered, roughly speaking, in three different basic formats (see [7] for a slightly different taxonomy).

structural: Documents in structural form contain information describing the content of the document, rather than prescribing the presentation of the information. For example, for text, instead of marking a section heading as 14 point bold followed by a 5 mm vertical space, it is declared as a level-2 heading. The same document may be rendered in different ways, depending on user preference and output device. This structural format also encourages consistency and allows automatic processing, e.g., the generation of tables of contents or indices. For example, a document might be viewed as a scroll rather than distinct pages or with larger fonts on a video terminal.

Examples of a structural format include SGML (Standard Generalized Markup Language) [8] and LaTeX [9] for text, input format for processors like pic, dot, and dag for graphics, HyTime [10] for multimedia documents with timing dependencies, or staff notation for sheet music. (Coombs *et al.* refer to this format as *descriptive markup*.) The typical storage requirements for text are about 2 to 5 kBytes per page, uncompressed, and less than half of that with Huffmann or Lempel-Ziv compression.

presentational: In the presentational format, the document specifies exactly where (and when, for time-based documents, such as musical performances) a character, a graphical element or a musical note is to be placed. It typically does not specify the lowest-level detail, e.g., exactly which pixels are to be shown black for a character. Presentational format has the advantage that it can abstract from the details of the device used for presentation.

For text and graphics, page description languages (PDL) are examples of this format, e.g., PostScript [11] or Acrobat [12, 13]. At a slightly higher level of abstraction, procedural markup languages like troff or TeX can be considered part of this group. MIDI and Apple QuickTime [14] could be considered the equivalents for music and movies, respectively. These types of documents are usually not directly editable, but some global manipulation such as scaling in time and space or font substitution may be possible. The storage requirements are modest, with around 10 to

20 kBytes per page uncompressed, and typically less than half of that with standard Lempel-Ziv compression.

bitmap: For text documents, pages can be represented as bitmaps, often stored compressed. Workstation monitors have a resolution of about 100 dpi (dots per inch), fax quality ranges from 200 to 400 dpi, low-cost laser and ink jet printers produce between 300 and 600 dpi, with typeset quality at around 1200 dpi. Standard digital audio and video also fall into this category. Uncompressed bilevel (black-and-white) images at 300 dpi require about 713 kByte of storage, reduced by a factor of ten or more through standard facsimile (Group 3 or 4) compression. For example, G4 compression reduces a typical technical report page to 15 kBytes [15].

Example applications include the AT&T RightPages [16, 17] and Ferret [15] systems which deliver bitmap images of scientific journals and AT&T internal technical reports, respectively, to networked workstations.

Both structural and presentational documents and, with a bit more difficulty, bitmap documents can contain links to other documents ("hypertext"). The World-Wide Web [18] is an example of a networked multimedia hypertext retrieval system using, for text, a particular form of SGML called Hyper-text Markup Language (HTML) [19]; other systems are described by Adie [1].

Among the formats described, a document in the structural format is most valuable to the subscriber, the publisher and the illicit user because it is easy to translate a document in this format to either the presentational or bitmap format (the translation in the other direction being hard if not impossible), and it requires the least storage space among the three formats. Moreover, a document in the structural format is device independent and is more flexible than the other formats. By the same arguments, a document in the bitmap format is the least valuable, and a document in the presentational format is less valuable than a document in the structural format and more valuable than a document in the bitmap format.

3 Requirements

In order to make electronic publishing a viable alternative to conventional publishing, the architecture and protocols that are developed need to meet the following requirements:

1. **No less secure than paper**

   Currently, users can photocopy, scan (i.e., convert to an electronic bitmap or, in limited ways, text) or fax documents. Electronic distribution of valuable documents will only gain widespread acceptance if the threat to the publisher's intellectual property is no greater than it is now. In

particular, large-scale distribution anonymous distribution over public data networks presents a new threat.

2. **Print and display**

   In many cases, the user wants to be able to view the document repeatedly. Also, the user may want to use the same document in different forms, such as printed for casual reading away from the workstation, or on a screen for convenience and hypertext capabilities. This implies that the system must work with different types of display devices (workstation, personal computer, X-terminal, etc.) and printers. This flexibility can either be accomplished by delivering a single document that can be read in different ways or by distributing the same basic content in different ways, appropriate to the intended use.

3. **Diverse network connectivity**

   The mechanisms should work for systems with direct data network connectivity, message-based systems (e-mail) and dial-up systems, possibly with different functionality.

4. **Guarantee of authenticity**

   The value added by the publisher includes authenticity and quality control; the user must be confident that he is indeed reading the genuine article and not a fake.

5. **Different levels of security**

   Depending on the value of the document, publishers may require different levels of security. For example, the publisher may insist on stringent security measures to prevent illicit copying of some valuable reference material like the Encyclopedia Britannica or specialized newsletters, but may require only moderate measures for copies of the daily local newspaper. Also, the security measures should not be such that there is loss of quality for honest users. For example, copy protection for most PC applications did not survive because of subtle hardware dependencies, and the inability to make backups of critical software.

   If a document has a sufficiently small audience, the publisher may decide that the likelihood of revenue-loss suffered by subscribers passing on documents to non-subscribers is sufficiently small that no cryptographic protection beyond access control is required.

6. **No large gain by reverse-engineering**

   Measures must be taken to prevent the mechanical use of a technique discovered by an expert to decrypt every encrypted document. It must be ensured that a technique which compromises one document does not compromise other documents. Thus, the cost of reverse-engineering needs to be made high enough and its payoff low enough to discourage people from attempting it.

7. **Publisher-independent**

   In hypertext applications a single document may contain links to documents from several different publishers. Even without hypertext, a user is likely to subscribe to journals from several publishers, and would find it rather inconvenient to switch applications and access methods just to read an article from a different journal. Thus, access methods and possibly encryption hardware devices or the application programming interfaces of software modules should be standardized.

8. **Dynamic and static documents**

   Traditionally, documents have been considered static, generated once and then stored for extended periods of time. However, there is no reason why dynamically generated documents, such as results from querying a database or the outcome of a computation, should not be distributed in a similar manner. This is already happening through the integration of various database front-ends into, for example, World-Wide Web servers. The user cannot readily discern whether a document existed as delivered or was created on demand. This requirement implies that any special processing necessary for copyright protection should have reasonable computational costs so that it is feasible even when performed on-line.

9. **Cacheable**

   It should be possible to store documents (in encrypted form) at sites other than the publisher, reducing the transmission cost and access delay. In particular, it offers an opportunity for distributed information service providers to serve as a storage center.

10. **Privacy**

    Libraries are very concerned that the nature and amount of the material borrowed by their patrons is not disclosed to third parties. Similarly, companies may have good reasons not to disclose to the world at large what kind of documents their research or legal staff is perusing. Thus, in an internetwork with physically insecure links, encryption for documents and requests in transit may well be a requirement even if no per-document charges are levied and redistribution is not a concern.

11. **Flexible billing**

    Different granularities of billing and access are required:

    by subscription: A professional society, for example, may bill its members at a flat rate, regardless of how many documents are retrieved by an individual. This pricing model reflects the low incremental costs of document access, encourages use, avoids refunds and contested charges, and incurs low billing overhead.

pay-per-document: A charge accrues only the first time a document is accessed, discouraging private storage of documents. If charges depend on the content or size of the document, there has to be a mechanism to ascertain the price before the actual document is retrieved or displayed. The document server has to store the identity of the retriever so that a second request can be recognized. Alternatively, the server could pass out a token to the user that this user can present to the publisher, proving that he indeed paid for the document.

pay-per-view: A charge is incurred each time a document is viewed or printed. This encourages local storage, if not disabled by appropriate cryptographic methods, but also reduces the amount of information stored at the server. This is the model followed by video rental stores.

Clearly, the same document may be accessible through different billing mechanisms.

12. **Independent of encryption algorithm**

    Algorithms should work with a range of encryption algorithms, as choices will often have to be made based on performance, export control or licensing considerations.

4 An Approach to Secure Document Distribution

In this section, we provide possible solutions to some of the problems raised in the previous section, without claiming that these are the only possible approaches.

We propose a two-pronged strategy to protect intellectual property: first, make it difficult for the rightful purchaser to distribute illegal copies. Two approaches are described in Section 4.2. Secondly, discourage such distribution by making the copies traceable to the original purchaser, with one solution described in Section 4.5.

The basic architecture that we propose for distribution of electronic documents has the following components:

Document server: provides encoded, encrypted and compressed document to user; trusted by the publisher.

Copyright server: authenticates requests from the user for obtaining documents; trusted by publisher.

Display client: software trusted by publisher; decrypts and displays document obtained from document server.

Printing client: software trusted by publisher; decrypts and prints document obtained from document server.

Network: transports document and requests.

User: generates request for document; may need to provide private key to display or print document.

Document access takes place in four phases as outlined below:

1. The user determines the unique identifier of the desired document through a bibliography database, a table of contents, or some other means. This phase is beyond the scope of this paper and probably does not require security measures beyond privacy, unless the existence of particular documents itself must be protected. A document could be identified by a universal resource name (URN) such as the international standard book or serial number (ISBN or ISSN) [20], which is then translated to the location of the document server and a file name, a universal resource locator (URL) [21], by a directory service such as X.500 [22] or whois++ [23].

2. The user requests the document by its URL and authenticates herself as described below. The request is encrypted to protect the user's privacy and signed to assure non-repudiation of the order. Alternatives for this step are described in Section 4.1.

3. The document is delivered to the user, encrypted to protect its content and the user's privacy. This and the next step are detailed in Section 4.2.

4. Software on the user's workstation displays or prints the document.

4.1 Access Control

Access to documents can be restricted in at least three different ways:

authentication: A user identifies himself to the server, allowing the server to ascertain that the user is authorized to obtain the requested document and that a billing relationship exists.

persona: It is envisioned that certification authorities might issue "persona" certificates that ensure anonymity, but allow for the registry of public keys. The details are described by Kent [24, p. 21].

anonymous: Analogous to anonymous ftp, the user does not establish an identity; payment is through some form of electronic cash or an anonymous credit card.

We briefly describe a number of alternatives for the case of authentication and personas, without distinguishing further between the two. The mechanisms are described here in roughly increasing level of security (and complexity):

User name and password: User name and encrypted password are sent in the clear. The password is encrypted using an algorithm like that used for Unix passwords. The scheme offers no protection against replay attacks.

Shared secret with time-dependent encryption: If the server has access to the unencrypted password, and both have clocks that are synchronized to within a few seconds, a more secure scheme is possible by changing the encryption based on the time of day at both client (user) and server. This scheme has the advantage of being stateless.

Challenge/response: The server issues a challenge, typically some number, to the client, which responds with the result of computing a function of that number, where the function is known to both client and server.

Kerberos: Kerberos offers password-based authentication using an authentication server [25, p. 430f]. No passwords travel across the network in plain text; time-stamps protect against replay attacks.

Public key: The client encrypts requests and the user's password with the server's public key. Since the number of servers is likely to be smaller than the number of users, the key distribution problem is somewhat simplified.

Public key certificates: The same algorithm as in privacy enhanced mail (PEM) [26] can be used. In that algorithm, the user encrypts a message digest of the request with his private key and includes a public-key certificate signed by a certification authority which includes the user's distinguished name and his public key.

Since the requests are typically short, the user could alternatively also send message and message encrypted with his private key, S_u, all encrypted with the public key of the copyright server, P_c:

$$P_c(\text{message}, S_u(\text{message})),$$

The copyright server can decrypt the message using its private key and the user's public key and compares the two replicated messages.

The choice of authentication method will largely depend on the implementation complexity and security requirements. In particular, the number of message exchanges should be minimized to decrease the amount of server state, the server's complexity and the retrieval delay*.

4.2 Document Delivery and Presentation

We propose two separate schemes for making electronic document distribution secure. Both protocols use the same basic architecture discussed above. The first protocol, described in Section 4.2.1, requires that a public key/private key pair, known only to the hardware manufacturer, be embedded in the

*The latter is particularly important if the document consists of a large number of individual files.

firmware of printing and display devices. In the second strategy, described in Section 4.2.2, the document is decrypted in software in the recipient's computer. Firmware modifications are not required, but the bitmap produced at the user's host may be available to the user and can be distributed.

In both approaches, we will make extensive use of asymmetric encryption schemes, that is, schemes where documents encrypted with a public key can be decrypted with a corresponding private key. Since asymmetric encryption is significantly slower than symmetric encryption methods such as DES, for all but very small messages only a symmetric session key is encrypted with the public or private key. The session key is then used to encrypt the actual message or document[†]. This method is implied where appropriate.

4.2.1 Firmware-Assisted Document Distribution

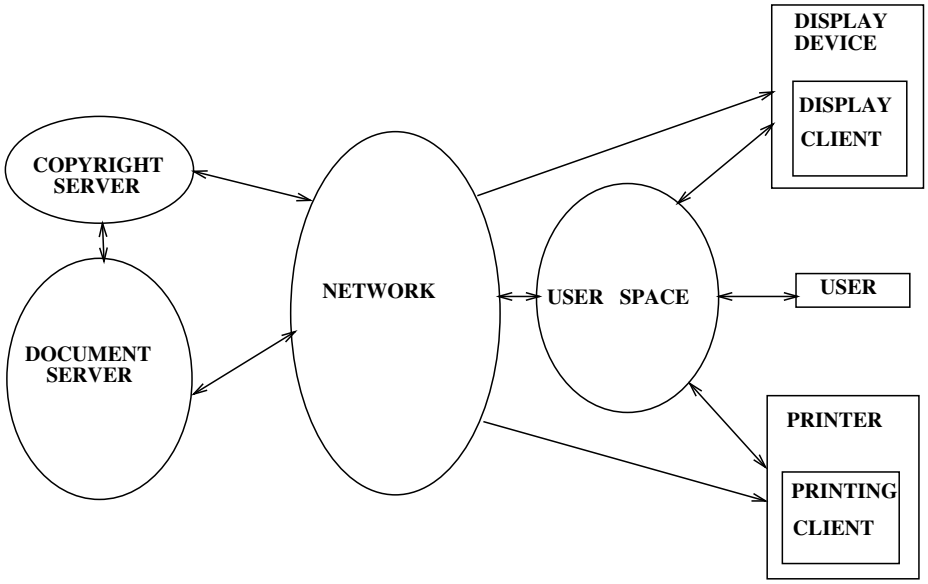

Figure 1: Architecture for electronic document distribution using firmware assistance

This approach involves a straightforward application of cryptographic techniques to send encrypted information between the document server and a display or printer trusted by the publisher (Fig. 1). The display or printing client contains a public key/private key pair. The device can be queried for its public key, which the document server uses to encrypt the document. Only the private key embedded in the device can then be used to decrypt the document. The protection relies on the fact that these devices cannot easily be coaxed into releasing anything but a paper copy or a bitmap image.

These extensions could easily be added to the firmware of an X-terminal or PostScript printer, without increasing hardware cost. The processors in most current PostScript printers and X devices

[†]This is the approach taken by privacy enhanced mail.

should be capable of performing decryption without significant performance penalty. High-throughput printers could add a DES chip to off-load the main processor. Protection of firmware against reverse engineering is a well-known issue with several solutions, in particular the integration of ROM and CPU.

The hardware manufacturer does not have to know about potential publishers. However, the public keys embedded in the devices have to be certified for printers or display devices of a type satisfactory to the publisher. Otherwise, a user could simply submit a public key belonging to a workstation program that decrypts the document and stores the decrypted version. Despite the necessity for certification, this approach is still preferable to hiding a secret key in the firmware. A single key for all printers and publishers requires only one dishonest employee at one manufacturer or publisher to break the whole system. Having every publisher contact every printer manufacturer for the list of secret keys also does not scale. Similarly, inserting a new, publisher-issued smart card containing the publisher's key is inconvenient for the user. Also, if somebody builds a smart-card reader and the appropriate software that mimics that found in the printer, the key can be extracted. Similar weaknesses argue against performing the decryption itself in a publisher-specific smart card or PCMIA card.

We believe that for certain high-value documents, this approach truly offers protection equivalent to paper documents. It may even be possible to display hypertext documents if the X protocol is enhanced in a manner similar to that suggested for displaying compressed video images.

4.2.2 Software-Based Document Distribution

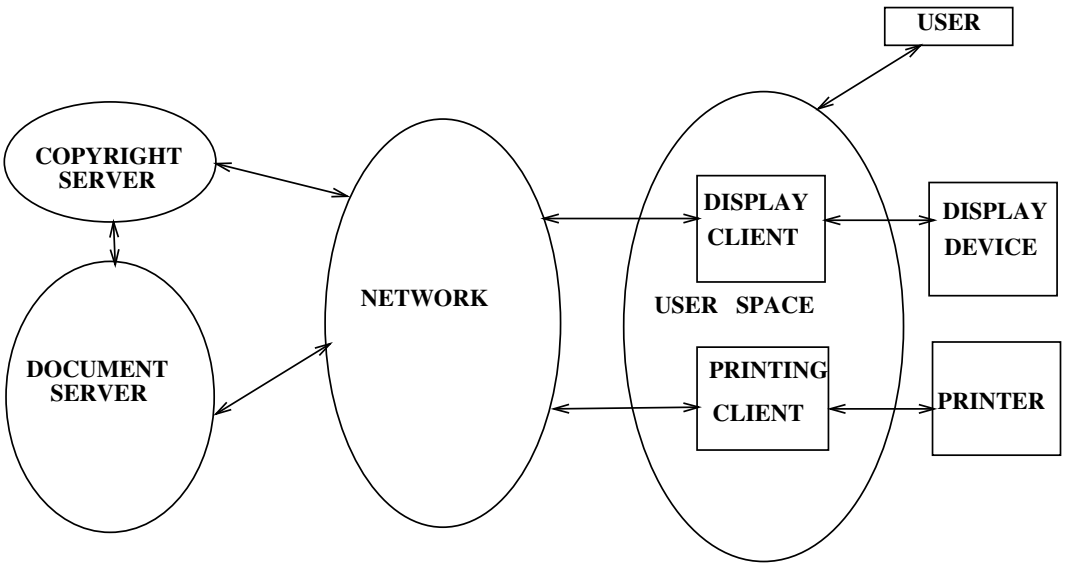

Figure 2: Architecture for electronic document distribution using software only

A more challenging approach uses cryptographic techniques that do not require special purpose firmware. The problem with conventional displays and printers is that the information that is displayed or printed exists in the recipient's computer. The recipient can capture the information that will be displayed, and can distribute that information to as many other printers and displays as desired. Instead of trying to prevent the recipient from redistributing information, our objective will be to discourage the distribution of illicit copies.

We outline some of the assumptions underlying this scheme and highlight its potential weaknesses.

1. **Display bitmaps are unprotectable**

   On a general-purpose workstation, it will always be possible to create a screen-dump, compress and distribute the resulting bitmap. For displays, the currently limited display resolution of around 100 dpi makes it less likely that screen dumps will be used for widespread distribution. (Laser printer resolution is 300 to 600 dpi.) Also, as pointed out earlier, bitmaps are less valuable than other forms of a document and require more memory, adding to transmission costs.

2. **Trusted user application**

   We will need an application residing at the user's workstation and completely under the user's control to cooperate in decrypting and displaying the document.

   Note that this is radically different from the typical problem of secure communication where a message is encrypted on a trusted machine, transmitted across a hostile domain, and finally decrypted on a second trusted machine. In this problem, the message is decrypted in the middle of the hostile domain. With sufficient resources and patience, it is possible to reverse-engineer the code, discover any built-in encryption keys, remove system calls used for authentication and thus defeat any such protection scheme. It is sufficient, however, to make the payback from any such reverse engineering smaller than the cost of the article. We take that approach and give the details later in the paper.

3. **User's private key is too valuable to give away**

   Documents that are delivered to the user are encrypted, and cannot be displayed or printed (as explained later) without the user's private key. Since an encrypted document is of no value to anyone other than the user holding the corresponding private key, it will not help the user to distribute an exact copy of what he obtained from the publisher unless he gives away his private key too. However, since the private key is something that could also be used for other applications like credit-card purchases, the user is unlikely to give it away.

Let d, c and u refer to the document server, the copyright server and the user, respectively. We assume that each user u has a pair of public and private keys P_u and S_u, respectively. Similarly, the

copyright server, c, has a pair of public and private keys P_c and S_c. In addition the document and copyright servers generate a key M_x which will be used to encrypt the transmitted documents. An encrypted version of M_x will also be embedded in the user-specific parts of the display and printing clients as described below.

After authentication (see Section 4.1), the copyright server sends the display client and the printing client to the user as message $m_1(c, u)$. The display client and the printing client have embedded in them a key $EP_u(M_x)$, i.e., the magic key M_x encrypted with the public key P_u of the user. The document server sends the encrypted document to the user as message $m_2(d, u)$.

$$m_1(c, u) = (\text{ display client}, \text{printing client })$$

$$m_2(d, u) = EM_x \text{ (document)}$$

To process a document, the client first prompts the user for his private key S_u and aborts unless provided with the correct one. The embedded key $EP_u(M_x)$ is decrypted with S_u to obtain the magic key M_x with which the compressed document is decrypted. This is further uncompressed, converted to a bitmap, possibly marked (see Section 4.5, and sent to the screen or printer.

The display and printing clients are not encrypted because nobody other than a specific user u can use them for decrypting an encrypted document. The document sent to the user is a compressed PDL version that is encrypted with M_x. Even if user u distributed the display or printing clients, together with the encrypted document, it would be of no use unless the private key S_u was divulged as well because the magic key M_x, with which the document is encrypted, cannot be generated from $EP_u(M_x)$ without S_u.

The above protocol will allow a legitimate user to request a document and view it on his workstation as many times as desired. However, it will prevent an illegal user from doing the same even if she happens to copy the display/printing clients and the encrypted document from the legal user. The underlying assumption in the protocol is that the user's private key, S_u, is too valuable for the user to give up. If the private key is the same as used for electronic mail signatures, system login or credit-card purchases, or even just allows the user to purchase other documents from the publisher, there is a strong disincentive to giving it away to others, presumably unethical "friends" to begin with. Also, we assume that it is non-trivial to tamper with the display/ printing client to discover the magic key M_x after the client decrypts $EP_u(M_x)$ with S_u.

Additional protection can be built in by restricting the document to be displayed or printed on some pre-registered hardware, for example, by checking for IP or MAC network addresses, CPU serial numbers or the presence of a hardware key ("dongle"). But this may not be desirable as it requires use of a specific host and requires each publisher to administer a host database, with possibly frequent updates.

Once the document has been decrypted and decompressed, it is available as a bitmap in the user's computer. Recall that (1) the bitmap is marked with information specific to user, u, and (2) the bitmap is larger than the original document transmitted by the publisher. So, even if the user is willing to capture and transmit the larger bitmap file, she can only do so at the risk of incriminating herself, unless she puts in the significantly larger effort required to erase the finger-print from the bitmap.

A more significant exposure occurs when displaying structured documents on X displays, using X fonts rather than bitmaps. The ability to use fonts has the great advantage that the display quality is typically higher, the user can easily customize the display to his tastes, and transmission over slow serial-line remote X servers is possible. Furthermore, short quotations for abstracts, reviews, survey papers, etc. can be easily extracted by standard X cut-and-paste or clipboard, without having to retype the document. (Since only a small part of the document is displayed at any given time, cutting and pasting together the whole would likely be only marginally faster than retyping the document.) If deemed necessary, it may also be possible for the display client to disable cut-and-paste either for the whole document or selected parts by appropriate X event bindings.

Another attack involves the X protocol itself. Since X allows for the separation of X server and X client among networked hosts, X packets containing bits and pieces of the document travel unencrypted across the user's network and could thus be retrieved and used to reassemble the document, given sufficient patience. Also, an X-server modified by the user could intercept the text X commands and thus reconstruct the displayed text. The publisher has to judge whether the likelihood of these activities outweighs the loss of functionality and speed incurred by only submitting bitmaps to the X server.

4.3 Printer Support

In order to prevent *wire tapping*, it makes sense to send an encrypted document rather than the plain bitmap to the printer from the user's machine. In that case, the printer needs to read the user's private key S_u before it can decrypt the document encrypted in P_u and print it. This requires additional processing capabilities for the otherwise dumb printers of today. With the printers available today, it will be possible to intercept the unencrypted bitmap sent to the printer. However, since the bitmap will be encoded with a unique signature identifying the original owner, the users will be reluctant to distribute the bitmap illegally.

4.4 Use-Once Programs as a Key-Hiding Mechanism

For the software-based approach, we require a program executing under user control to display and print documents. This program has to be trusted by the publisher. For example, in our protocol, there

is a display (printing) client program which conceals the publisher's hidden key. Note that if the user can discover M_x by analyzing the code for the client, then the whole purpose of sending encrypted documents is defeated. Since this kind of reverse engineering cannot be completely prevented, we aim to reduce the payoff of reverse engineering by sending trusted programs that do the same job but look different for each user. If the documents and the client programs are distributed through networks, it is relatively easy to generate a unique copy for each recipient.

We envision three different levels of security:

1. All users have the same display or printing client containing an algorithm that derives a key from a system identifier such as a MAC or network-layer address known to both server and client.

2. The display or printing client are sent once (or at some time interval but unique for each user.

3. The display/printing client are transmitted with each article.

Clearly, placing the hidden key at a fixed program location or surrounded by a tell-tale pattern of code makes it trivial to write a "universal descrambler" that can extract the key from every single client. Also, programs that always fetch the key after a fixed number of instructions are subject to easy debugging. A number of techniques can be used for creating unique, but equivalent programs automatically at the compile or link stage.

- the definition of the hidden key variable can be placed into different modules;
- the linker can reorder text and data segments;
- the compiler can be instructed to randomly optimize certain sections of code;
- sections of the code can be replaced by functionally equivalent, but different algorithms
- the compiler can change the register allocation sequence

Note that more elaborate schemes for protecting RAM access patterns and contents may be added, at the cost of reduced execution efficiency [27]. In addition, standard techniques of hiding tell-tale code sequences (such as replacing system calls to constant addresses by computed calls) should be used. Also, it may be helpful to use a slight variation on a standard encryption method so that the whole decryption engine has to be disassembled.

We can also make reverse engineering more difficult by having the client send a message to the server notifying it when it has completed the printing or display process, i.e., after the decryption has been performed. If the user is employing a debugger to step through the decryption code, the time between retrieving the document and the "completed" message would be significantly longer, tipping

off the server that a potential infringement is occurring. Only with sophisticated hardware could the instruction sequence be reconstructed without delaying execution.

4.5 Origin Tracing for Documents and Programs

As described by Brassil *et al.* [6], bitmaps can be marked differently for each recipient. The marking can be done in such a way that it is difficult for a dishonest user to discover the unique marking pattern used in the user's document by direct comparison of two legitimate copies. For text rendered as bitmaps, line and word spacing can be imperceptibly changed or the shape of letters subtly modified. Similarly, it may be possible to derive functionally equivalent, but syntactically different representations of a PDL or SGML document. It has to be sufficiently difficult to "neuter" the document by transforming it into a new and unidentifiable document. For example, simply adding invisible blanks or extra line feeds could easily be defeated by mechanically stripping all unnecessary white space.

If the document is transmitted either in structural or presentational format, the markings can be added either by the server or the client. If the server adds it, less sensitive information is exposed in potentially vulnerable clients, but the server requires more CPU cycles to generate a custom documents.

Since each use-once display or print client is different, tracing their origins is not very difficult. Furthermore, modifying binaries to create another working copy is rather difficult once they have been "stripped", that is, symbolic debugging information has been removed. Client programs could either be identified by a hash checksum or contain a suitably hidden identification tag. Compared to identifying printed articles, identifying programs is relatively easy.

5 Conclusions

The advantages of electronic publishing, namely lower costs, greater flexibility in presentation and the ease of distribution, will make electronic publishing a reality once the problem of distributing illegal copies of an original document is solved. In this paper, we presented an architecture and a protocol which makes the distribution of illegal copies difficult. The key idea in our solution is to use a display (or printing) client which is responsible for decrypting the encrypted document sent to the intended user. These clients have embedded in them a key. Upon execution, the application uses the user's private key (provided as an input) to access the key, which is then used to decrypt the delivered document. Thus, distributing the clients serves no purpose without distributing the private key of the user as well. We admit that this technique is not foolproof because the display and printing clients reside in the user space; if the user is sophisticated, he can break into the system with possibly a considerable effort. However, if the same concept is extended to a future system in which the display (printing) client can be sealed inside a special-purpose display device (or printer) instead of being

kept as a piece of software in accessible user space, then the system can be made secure. We believe that as publishers realize the potential of electronic publishing, there will be an incentive to add the necessary firmware to printers, displays and set-top boxes to physically secure the system.

References

[1] C. Adie, "Network access to multimedia information," Internet Draft, Edinburgh University Computing Service, Nov. 1993. Work in progress.

[2] J. R. Garrett, "Text to screen revisited: copyright in the electronic age," *Online*, vol. 15, pp. 22–24, Mar. 1991.

[3] A. E. Luchsinger, "The future of scientific communications: more questions than answers," *Annals Missouri Botanical Garden Society*, vol. 80, pp. 297–303, 1993.

[4] S. C. Loken, R. J. Beach, K. T. McDonald, T. D. Schultz, H. Thacker, and J. Worth, "Report of the APS task force on electronic information systems," *Bulletin of the American Physical Society*, vol. 36, pp. 1119–1151, Apr. 1991.

[5] N. R. Wagner, "Fingerprinting," in *Proceedings of the 1983 Symposium on Security and Privacy*, (Oakland, California), pp. 18–22, IEEE, Apr. 1983.

[6] J. T. Brassil, S. Low, N. F. Maxemchuk and L. O'Gorman, "Electronic Marking and Identification Techniques to Discourage Document Copying", in *Proceedings of IEEE INFOCOM '94*, (Toronto, Canada), pp. 1278-1287, IEEE, June 1994.

[7] J. H. Coombs, A. H. Renear, and S. J. DeRose, "Markup systems and the future of scholarly text processing," *Communications ACM*, vol. 30, pp. 933–947, Nov. 1987.

[8] International Standards Organization, *Standard Generalized Markup Language (SGML)*. Geneva, Switzerland: International Standards Organization, 1986.

[9] L. Lamport, *LaTeX: A Document Preparation System*. Reading, Massachusetts: Addison-Wesley, 1986.

[10] International Standards Organization, *ISO 10744: Hypermedia/Time-based Structuring Language*. Geneva, Switzerland: International Standards Organization, 1993.

[11] Adobe, *PostScript language reference manual*. Reading, Massachusetts: Addison-Wesley, 2nd ed., 1990.

[12] Anonymous, "Carousel becomes Acrobat," *Electronic Documents*, vol. 1, pp. 7–8, Dec. 1992.

[13] Adobe, *The Portable Document Format Reference Manual*. Reading, Massachusetts: Addison-Wesley, 1993.

[14] Apple Computer, *Inside Macintosh: QuickTime*. Apple Computer, Cupertino, California, 1.5 ed., Oct. 1992.

[15] H. H. Katseff and T. B. London, "The Ferret document browser," in *Proceedings of USENIX Summer 1993 Technical Conference*, (Cincinnati, Ohio), pp. 101–110, USENIX, June 1993.

[16] G. A. Story, L. O'Gorman, D. Fox, L. Schaper, and H. V. Jagadish, "The RightPages image-based electronic library for alerting and browsing," *IEEE Computer*, vol. 25, pp. 17–26, Sept. 1992.

[17] L. O'Gorman, "Image and document processing techniques for the RightPages electronic library system," in *International Conference on Pattern Recognition (ICPR)*, (The Hague, Netherlands), pp. 260–263, Sept. 1992.

[18] T. Berners-Lee, R. Caillian, and J.-F. Groff, "The world-wide web," *Computer Networks and ISDN Systems*, vol. 25, pp. 454–459, Dec. 1992.

[19] T. Berners-Lee and D. Connolly, "Hypertext markup language," Internet Draft, CERN, June 1993. Work in progress.

[20] C. Weider and P. Deutsch, "Uniform resource names," Internet Draft, Merit Network, Oct. 1993. Work in progress.

[21] T. Berners-Lee, "Uniform resource locators," Internet Draft, CERN, July 1993. Work in progress.

[22] M. T. Rose, *The Little Black Book: Mail-Bonding with OSI Directory Services*. Englewood Cliffs, New Jersey: Prentice-Hall, 1991.

[23] C. Weider, J. Fullton, and S. Spero, "Architecture of the whois++ index service," Internet Draft, Merit Network, Oct. 1993. Work in progress.

[24] S. Kent, "Privacy enhancement for internet electronic mail: Part II: Certificate-based key management," Network Working Group Request for Comments RFC 1422, IETF, Feb. 1993.

[25] W. R. Stevens, *UNIX Network Programming*. Englewood Cliffs, New Jersey: Prentice-Hall, 1990.

[26] J. Linn, "Privacy enhancement for internet electronic mail: part I: message encryption and authentication procedures," Network Working Group Request for Comments RFC 1421, IAB IRTF PSRG, IETF PEM WG, Feb. 1993.

[27] R. Ostrovsky, "An efficient software protection scheme," in *Advances in Cryptology — CRYPTO '89* (G. Brassard, ed.), vol. 435, (Santa Barbara, California), pp. 610–611, International Association for Cryptologic Research, Springer Verlag, Aug. 1989. Lecture Notes in Computer Science.