Render Sequence Encoding for Document Protection

Baoshi Zhu, Jiankang Wu, and Mohan S. Kankanhalli, *Member, IEEE*

***Abstract*—We present in this paper a novel electronic document watermarking method, render sequence encoding (RSE), and then further develop a RSE authentication method for electronic documents. RSE watermarks an electronic document by modulating the display sequences of words or characters. It features large information-carrying capacity and robustness over document format transcoding. The RSE authentication method is based on the NP-complete Exact Traveling Salesman Problem, which provides a rigorous foundation for security. The RSE authentication method is secure in the sense it is extremely difficult to forge the authentication process. RSE authentication process is also easy to operate, especially in comparison to digital signatures which requires Public Key Infrastructure for its operation.**

***Index Terms*—Authentication, Exact Travel of Salesman Problem (XTSP), render sequence encoding, tamper detection.**

I. Introduction

RESEARCH activities in the digital rights management for electronic documents have been growing due to its commercial potential. The digital rights management (DRM) market for electronic documents will have tremendous growth as forecasted by both academic and business societies. However, adoption of electronic documents into serious business and administrative transactions is still limited due to the unavailability of effective means for managing access rights, content integrity and authenticity.

Authenticating electronic documents has been a subject of research in both the cryptography and multimedia communities. Digital signature is a classical and the most effective method for electronic document authentication [1]. However, it does need the Public Key Infrastructure (PKI), which may not be available in many business communities and administrative environments.

Unlike digital signature, which protects the binary codes of documents, digital watermarking is regarded as a type of content-based authentication method, which protects the visual content of documents. Digital watermarking research mainly focuses on protection of images, video and audio [2]. It considers the media content as a random digital signal, and embeds authentication information into the "busy" parts of the content. In case of electronic documents, the content usually does not show much randomness. In most cases, electronic documents are in a structured text form (such as XML document, and PDF files) and not the raster image format. As such, digital watermarking techniques for images and video are not directly applicable to electronic documents. Existing watermark schemes for electronic documents try to modify the layout or appearance of the underlying electronic document. These schemes are to change: 1) line spacing; 2) word spacing; 3) character spacing; 4) the shape of font; and 5) blank spaces of those chosen lines, words, and characters, respectively [3]–[11].

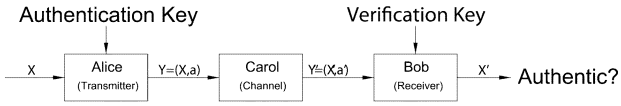


Fig. 1. Authentication model.

Fig. 1 [12] is the general authentication model. In the model, Alice is the signing authority of a document X. With a given authentication key and method, she signs document X, and sends the signed document Y to Bob via the transmission channel (Carol). After receiving the document Y, Bob or related parties can verify the authenticity of the document by retrieving the verification key from an authentic database and published method.

In case of digital signatures, the authentication key is Alice's private key, while the verification key is Alice's public key. The method used in the authentication model is the hashing algorithm which is common to both signing and verification operations. The framework is ideal for the protection of the integrity and authenticity of digital document in a binary form. Digital signature fully relies on PKI, which so far is not widely available due to its social and technical sophistication and complexity.

The idea of digital watermarking is to protect the digital content, rather than binary form. It can survive certain trans-coding operations. For the given authentication information, parameters for embedding the authentication information into the given document are generated. After the embedding process (or signing, in other words), the authentication information and/or embedding parameters are sent to the trusted database, which can be later retrieved and used by the receiver and other related parties to verify the authenticity of the document. Differing from digital signature, where verification key is public key, authentication information and embedding parameters are secret in the authentication model. In the existing document watermarking techniques mentioned above, there is no protection at all when the authentication information and embedding parameters are known to attackers or public.

We propose a novel electronic document authentication method based on render sequence encoding (RSE). It has merits of both digital signatures and digital watermarking: like digital signature, with verification key/information, one

Manuscript received March 4, 2005; revised April 17, 2006. The associate editor coordinating the review of this paper and approving it for publication was Prof. Tihao Chiang.

B. Zhu is with Trustcopy Pte Ltd., Singapore Technologies Building, Singapore 088934 (e-mail: baoshi.zhu@gmail.com).

J. Wu is with the Institute for Infocomm Research, Singapore (e-mail: jiankangwu@gmail.com).

M. S. Kankanhalli is with the Department of Computer Science, School of Computing, National University of Singapore, Singapore 117543 (e-mail: mohan@comp.nus.edu.sg).

Digital Object Identifier 10.1109/TMM.2006.886334

can perform verification, but not (or with great difficulties to) reverse engineer the signing process; it provides content protection, and survives trans-coding processes.

Unlike the existing digital watermarking methods which hide information into pure text documents or image based documents, RSE hides information into formatted documents. Those formatted documents contains both text data and layout information. Typical ones are PostScript (PS®[1]), Portable Document Format (PDF), Printer Control Language (PCL®[2]), Device Independent Document (DVI), etc. Formatted document combines the advantages of both text document and image based document. They have small file size and platform-neutral page layout. It is widely used in electronic publishing, business, and administrative processesing.

RSE, as the name indicates, encodes hidden data into the layout information of the formatted documents by modifying the display sequences of words or characters, without changing the content or appearance of the document. The encoding is achieved by specific permutation of display sequence, and the specific permutation is further linked to a Hamiltonian tour T in the *Exact Travel of Salesman Problem (XTSP)*. It needs a properly chosen cost matrix C for the XTSP, and then the cost of the tour is made modulo equal to the content of digest L. Then, T, C, and L are modeled by XTSP. Because of the hardness of XTSP, with published C, one cannot forge the signing process and attack the RSE authentication method.

The paper is organized as follows. Section II describes the fundamentals of our document watermarking method. In Section III, we propose an authentication scheme for electronic documents. The conclusions are presented in Section IV.

II. Render Sequence Encoding

RSE is a watermarking scheme and authentication method for electronic documents. The prominent feature which distinguishes RSE from digital signature is that it uses content-based watermark to embed the authentication information into the document such that the information becomes an inseparable part of the document. It withstands document format trans-coding—the interoperability issue in many applications, where authenticity and access rights concerning a specific document must be maintained across different document formats. In this section, we present the RSE watermark scheme by first explaining its basic idea using a simple example, then describing its encoding and decoding algorithms, and finally discussing its information-carrying capacity and robustness against format transcoding.

A. Basic Idea of RSE

Unlike other document watermarking schemes which embed payload data into pure text documents or image-based documents, RSE embeds data into formatted documents. Formatted document refers to the document format which contains both text data and layout information. It is also called a vector-based document format so as to distinguish it from the image-based document formats. Formatted document combines the advantages of both text documents and image-based documents: the small file size and platform independent page layout. In formatted documents, text data are described using 3-tuples of *(character, font, position)*. For example, Fig. 2 is a simple document which uses the PostScript language to describe the sentence "`This is a sentence.`" Line 1 moves the cursor to the target position, and line 2 draws the character string. Note that there is only one positioning command "`100 200 moveto`", which defines the position of the first character "`T`". All of the remaining characters are advanced horizontally according to the character width stored in the font definition. The description of the 3-tuples are very clear and succinct. However, such simple formatting methods are rarely used in practice. Word processing software packages usually issue several positioning commands for a single text line, in order to satisfy the requirements on text justification and font kerning (adjustment of space between pairs of letters to make them more visually appealing). For example, the above sentence is formatted as Fig. 3 by the LATEX document preparation system (we have expanded all PostScript macros to make the code more readable). In this real sample, the sentence has been split into five code segments (line 1–2, 3–4, 5–6, 7–8, 9–10), each consists of a positioning command and a drawing command. Examining the five positioning commands, we can find that the position parameters they took are sorted in the normal reading direction, that is, from left to right. We now randomly permute the five segments so that the position parameters are no longer sorted, as shown in Fig. 4. Obviously, Fig. 4 has exactly the same appearance as Fig. 3 after being rendered, but at the binary level they are different. In fact, we can create up to $5! = 120$ visually identical documents by using different permutations in the permuting of the five segments. From variance-tolerance perspective, the variant permutations among all documents are tolerable by the computer's rendering system. This predicates the existence of redundancies in document description. The particular form that redundant information takes here is the sequence of positioning and corresponding drawing commands.

```
1  100 200 moveto              % Positioning
2  (This is a sentence.) show  % Characters
3  showpage
```

Fig. 2. Simple PostScript document.

```
 1  100 200 moveto              % Positioning
 2  (This) show                 % Characters
 3  124 200 moveto              % Positioning
 4  (is) show                   % Characters
 5  135 200 moveto              % Positioning
 6  (a) show                    % Characters
 7  144 200 moveto              % Positioning
 8  (sentence) show             % Characters
 9  185 200 moveto              % Positioning
10  (.) show                    % Characters
11  showpage
```

Fig. 3. PostScript document with explicit positioning Commands.

[1]PostScript is a registered trademark of Adobe Systems, Inc.

[2]PCL is a registered trademark of Hewlett-Packard Company.

```
 1  124 200 moveto              % Positioning
 2  (is) show                   % Characters
 3  144 200 moveto              % Positioning
 4  (sentence) show             % Characters
 5  100 200 moveto              % Positioning
 6  (This) show                 % Characters
 7  185 200 moveto              % Positioning
 8  (.) show                    % Characters
 9  135 200 moveto              % Positioning
10  (a) show                    % Characters
11  showpage
```

Fig. 4. Randomly permuted Postscript document.

When the formatted document get rendered either on the computer screen or on the printer, the render engines will preserve the permutation of positioning and drawing commands by interpreting the commands according to their storage order in the file, rather than resort all of the commands. This is because of the following.

1) Most page description languages, e.g., PostScript, are based on the imaging model that "Each new mark completely obscures any marks it may overlay" [13]. Sorting the order of drawing commands directly changes the overlay structure among image objects hence changes the whole document appearance. So without knowing whether two objects are actually overlapping, the render engine cannot change the order of commands.
2) If otherwise the engine chooses to re-sort the order, it must interpret and rasterize all the drawing commands to examine the overlay structure of all image objects. Such practice is very inefficient and clearly will significantly increase the cost of memory and processing.

As a result, when a document like Fig. 4 is rendered, the word "`sen`" shows up first, followed by "`tence.`", "`is`", "`This`" and "`a`". The render sequence no longer follows the normal reading direction. Therefore, we use the *RSE* name for our watermarking scheme. Generating a unique permutation of render sequence based on the payload data forms the basis of our RSE scheme.

To create more redundancies, as well as to facilitate encoding and decoding efficiency, the permutation of positioning commands can be applied to characters instead of words, e.g., permuting the positions of each character "e" in a whole page. We call the character being permuted the *permutation target*.

B. Encoding Algorithm

RSE embeds information into the electronic document by permuting the render sequence according to a certain permutation. The encoding algorithm generates such a permutation based on the payload data. Two problems to be tackled are: 1) how to identify a permutation and 2) how to map payload data to the identification of a permutation.

An obvious method (called *normal notation* hereinafter) to identify a permutation is to list the occurrence of each element directly, for example, the notation $\langle 3,4,1,5,2\rangle$ means element 3 is permuted to the first place while 2 to the last. However, there is significant dependence in the choice of each element. Once an element has been used, it cannot appear in the following sequence again. This issue creates some difficulties in enumerating all permutations if the number of elements is large. A work-around is to find a way to list all permutations sequentially, so that we can identify each permutation using its index. An algorithm for generating all permutations sequentially based on normal notation was discovered by Johnson [14] and Trotter [15] independently, and was described by Gardner in [16]. The algorithm is quite simple, but in order to determine the ith permutation, all $i-1$ permutations must be generated first, which will be time consuming for large permutation length, hence also unacceptable.

Here we use another method called *inversion notation* to identify a permutation, and introduce a fast algorithm to map arbitrary payload data to an inversion notation. Describing a permutation by means of its inversion notation was discovered by Hall [17].

Definition 2.1: Let $\langle i_1, i_2, \ldots, i_n\rangle$ be a normal notation of a permutation in the set $\{1,2,\ldots,n\}$. The pair (i_j, i_k) is called an *inversion* if $j<k$ and $i_j > i_k$.

For example, the permutation $\langle 3,4,1,5,2\rangle$ has five inversions: (3,1), (3,2), (4,1), (4,2), and (5,2).

Definition 2.2: For a permutation $\langle i_1, i_2, \ldots, i_n\langle$, let a_j denote the number of inversions whose second component is j. In other words, a_j is equal to the number of integers which precede j in the permutation but are larger than j; it measures how much is j out of order.

The sequence of numbers $\langle a_1, a_2, \ldots, a_n\rangle$ is called the *inversion sequence* of the permutation $\langle i_1, i_2, \ldots, i_n\rangle$.

For example, the inversion sequence for permutation $\langle 3,4,1,5,2\rangle$ is $\langle 2,3,0,0,0\rangle$.

Theorem 2.1: The inversion sequence $\langle a_1, a_2, \ldots, a_n\rangle$ of the permutation $\langle i_1, i_2, \ldots, i_n\rangle$ satisfies this condition

$$0 \le a_1 \le n-1, 0 \le a_2 \le n-2, \ldots, 0 \le a_{n-1} \le 1, \quad a_n = 0.$$

This is because for each $k = 1,2,\ldots,n$ there are $n-k$ integers in the set $\{1,2,\ldots,n\}$ which are larger than k. Brualdi in [18] shows that the mapping between normal notation and inversion notation is onto, and gives conversion algorithms between these two notations. The advantage of using inversion notation is that we can choose each element in an inversion sequence independently, as long as Theorem 2.1 is satisfied.

Procedures for mapping arbitrary payload data into inversion sequence are as follows.

Algorithm 2.1 (Encoding Algorithm): (Given payload data, output a unique corresponding inversion sequence) For any integer X:

1) choose an appropriate number n such that $n! > X$;
2) calculate

$$\begin{array}{ll} a_1 = X \div (n-1)! & X_1 = X - a_1 \times (n-1)! \\ \cdots & \cdots \\ a_k = X_{k-1} \div (n-k)! & X_k = X_{k-1} - a_k \times (n-k)! \\ \cdots & \cdots \\ a_{n-1} = X_{n-2} & X_{n-1} = X_{n-2} - a_{n-1} = 0 \\ a_n = X_{n-1} = 0 & \end{array} \tag{1}$$

where $a \div b = \lfloor a/b \rfloor$, the integer part of (a/b);

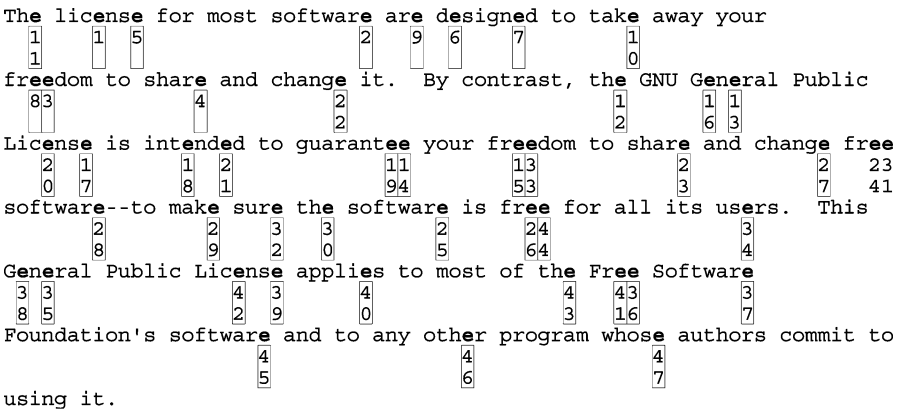


Fig. 5. Sample encoded document.

3) sequence $\langle a_1, a_2, \ldots, a_n \rangle$ is the inversion sequence identified by data X.

We omit the proof of RSE encoding algorithm as the result is easily derived.

By now we have solved the two problems raised at the beginning of this section: 1) a permutation can be uniquely identified using its inversion sequence and 2) the encoding algorithm can uniquely map the payload data to an inversion sequence.

Fig. 5 illustrates a sample RSE encoded document[3]. We choose string "GNU" as the payload and $n = 11$ as the permutation length. By applying Algorithm 2.1, we get the inversion sequence of the desired permutation as $\langle 1, 2, 7, 7, 1, 2, 2, 3, 1, 1, 0 \rangle$ and the corresponding normal notation $\langle 11, 1, 5, 2, 9, 6, 7, 10, 8, 3, 4 \rangle$. We choose character "e" as the permutation target. Within the total 47 "e"s in the document, we can perform four rounds of permutations. As seen in Fig. 5, the position of the first character "e" in the encoded document is actually the position of the second "e" in the original document, that of the seconds is the fourth in the original document, and so on.

C. Decoding Algorithm

To decode a RSE encoded document, we must first extract the permutation from the document, then decode the permutation to recover the payload X. The procedures are as follows.

Algorithm 2.2 (Decoding Algorithm): (Given an RSE encoded document, output the encoded payload data X) Given any RSE encoded document, do the following steps.

1) If the permutation target and permutation length n is known, go to step 6, otherwise go to step 2.
2) Find the permutation target by examining the positioning commands for each character.
3) Record all the positions for the examined permutation target and discover the render sequence. It is done by comparing the physical storage order of these positions with the logical positions they represent. The sequence is denoted $\langle S_1, S_2, \ldots, S_m \rangle$.
4) Generate a new sequence S' such that $S'_k = S_{k+1} - S_k$.
5) Find the largest period of the sequence S', it is the permutation length n.
6) Discover the actual render sequence $\langle a_1, a_2, \ldots, a_n \rangle$ with the permutation target and the permutation length n.
7) Calculate the encoded payload X:

$$X = \sum_{i=1}^{n} a_i \times (n-i)! \tag{2}$$

Equation (2) holds because it is just the reverse form of (1).

It must specially noted that in order to carry out steps 2–5 to determine the permutation target and length, several prerequisites must be satisfied, including: 1) the decoder must know how to distinguish permutation targets, though it may not know which exact permutation target is used. For example, it is possible to permute character "e" to encode one message, and permute character "a" to encode another. Then the decoder must know that the permutation target contains one single character and 2) the permutation used must not contain repeatable "subpermutations". For example, the permutation $\langle 3, 1, 2, 6, 4, 5 \rangle$ (normal notation) is not allowed, because it will be treated as concatenation of permutation $\langle 3, 1, 2 \rangle$ in step 5. These two prerequisites, especially the second one, force some limitations on the RSE scheme. However, for authentication purpose the permutation target and the permutation length can be made public, transferred through auxiliary channels, or pre-agreed between the encoder and the decoder. Publicizing this information does not prevent the RSE scheme from being a public watermark scheme, because they contain no information about the original document. Nevertheless, steps 2–5 can still be used as backup mechanisms in case the extraction of permutation target and length is needed. We use the previous RSE encoding example to see how the encoded information is decoded.

We first determine the permutation target is character "e" because its positions are in abnormal order. By comparing the storage order of the positioning commands with the positions they represent, we have $\langle S_1, S_2, \ldots, S_{47} \rangle = \langle 11, 1, \ldots, 46, 47 \rangle$ and by step 4, we get

$$\begin{aligned}\langle S'_1, \ldots, S'_{46} \rangle = \langle & 10, -4, 3, -7, 3, -1, -3, 2, 5, -1, -18, \\ & 10, -4, 3, -7, 3, -1, -3, 2, 5, -1, -18, \\ & 10, -4, 3, -7, 3, -1, -3, 2, 5, -1, -18, \\ & 10, -4, 3, -7, 3, -1, -3, 2, 5, -1, -8, \\ & -1, -1 \rangle.\end{aligned}$$

[3]http://www.gnu.org/copyleft/gpl.html

TABLE I
FILE SIZE AND ENCODED BITS VERSUS PERMUTED CHARACTERS

Choice of Permutation targets	Number of Permutation targets	Number of encoded bits	File size (ps.bz2, bytes)
{ø}	0	0	1701
{e}	284	1910	3121
{e,t}	511	3866	4026
{e,t,o}	710	5706	4756
{e,t,o,r}	883	7373	5144
{e,...,i}	1037	8899	5601
{e,...,a}	1187	10416	5947
{e,...,s}	1338	11972	5844
{e,...,n}	1466	13310	6082
{e,...,h}	1555	14250	6198
{e,...,u}	1628	15026	6271

The largest period of the sequence S' is 11, so the encoding is on an 11-permutation (note that the last two lines of S' are exceptions, because encoding had been truncated). We then obtain the real permutation by examine the first 11 "e"s, which is $\langle 11,1,5,2,9,6,7,10,8,3,4\rangle$, and the corresponding inversion sequence is $\langle a_1,a_2,\ldots,a_n\rangle = \langle 1,2,7,7,1,2,2,3,1,1,0\rangle$. Apply step 7, we get $X = 4673109$, the ASCII representation of "GNU".

D. Information Carrying Capacity

RSE is based on the permutation of the render sequence. The length of the permutation determines how many different permutations can be generated, and hence determines the information carrying capacity. For a page of document containing n permutation targets, the maximum length of permutation is n, and the maximum number of permutations can be generate is $n!$ (n's factorial). The number of bits that can be encoded using n-permutation is: $N_{\text{bits}} = \lfloor \log_2 n! \rfloor$. For example, a 15-permutation is capable of carrying 40-bit of information.

It is expected that, by adding positioning commands into the original document, the file size will be increased. We use experiments to show the effect of size increase versus the number of permutation targets. This experiment takes a page of pure text data as input, outputs two PostScript documents, one unencoded, and the other encoded with selectable permutation targets. We choose up to the ten most frequently used characters in the original text as the permutation targets (in the experiment, $\mathrm{e,t,o,r,i,a,s,n,h,u}$). Table I shows the relationship among the choice of permutation targets, the number of permutation targets, the encoded bits and the file size in "ps.bz2 (PostScript with bzip2 compression)" format. We find the addition of permutation targets dramatically increases the information carrying capacity of RSE, but the enlargement of file size is comparatively small.

E. Robustness

An important requirement on the watermark scheme for authentication purpose is the robustness against document format transcoding. This property helps to ensure document security in an interoperable environment. We use experiments to show RSE scheme satisfies this requirement.

In the experimental setup, we create a virtual printer driver which stores the position of printed characters into a local file instead of printing onto a real printer. The printer driver is programmed as a Ghostscript [4] device, and serves as the back end of the Common Unix Printing System (CUPS) [5] printing system. Through this architecture, all document formats other than Postscript are converted into Postscript, and the permuted render sequence is captured at the virtual printer driver.

The encoded source document is in the Postscript format. We convert it into the PDF format (using ps2pdf13 and Adobe Distiller respectively), PCL format (using ljet4 with Ghostscript), PCLXL format (through a HP Windows printer driver), and EPS format (using epswrite with Ghostscript). We then send the converted files into the CUPS system and examine the layout of the permutation targets. We have performed the same experiments for 50 different encoded source file with distinct permutation targets. All of them preserve the render sequence successfully.

However, RSE only targets formatted document as the host signal, the hidden information is totally removed if the formatted document is converted into other types of documents, e.g., image based documents such as TIFF of JPG. If such conversion is inevitable in the document workflow, we would suggest the use of other image watermarking methods or document watermarking methods which modify document layout or appearance.

In another experiment, we create an encoded Postscript document, edit it using the Adobe Illustrator by changing several characters, then save it back into a PDF file. When the edited PDF file gets printed, we discover that the encoded information at the places where changes are made has been destroyed, while the encoded information at the unchanged places is preserved perfectly. This property shows the fragility of RSE against modifications. It enables a tamper detection application without access to the original version.

F. Tamper Detection With RSE

In Section II-E, we use experiments to show the fragility of RSE against modifications. Modifying a RSE encoded document will only destroy the embedded information at the places where modifications are made. We encode the same permutation into the document for multiple rounds, or onto multiple permutation targets, then the modified places can be located by comparing the permutations. Fig. 6 shows an example of tamper detection with RSE.

The sample document has been modified by deleting words "and change" in line 3. As a result, the 27th letter "e" is missing and all "e"s after it are shifted forward by one position. Here, we consider the situation that we do not know the original render sequence. (If, on the contrary, we know the original render sequence, then detecting modified places is much easier.) In Fig. 6, the render sequence is

$$\begin{aligned}\langle &11,1,5,2,9,6,7,10,8,3,4,\\ &22,12,16,13,20,17,18,21,19,14,15,\\ &32,23,24,31,27,28,31,29,25,26,\\ &43,33,37,34,41,38,39,42,40,35,36,\\ &44,45,46\rangle.\end{aligned}$$

[4] http://www.ghostscript.com

[5] http://www.cups.org

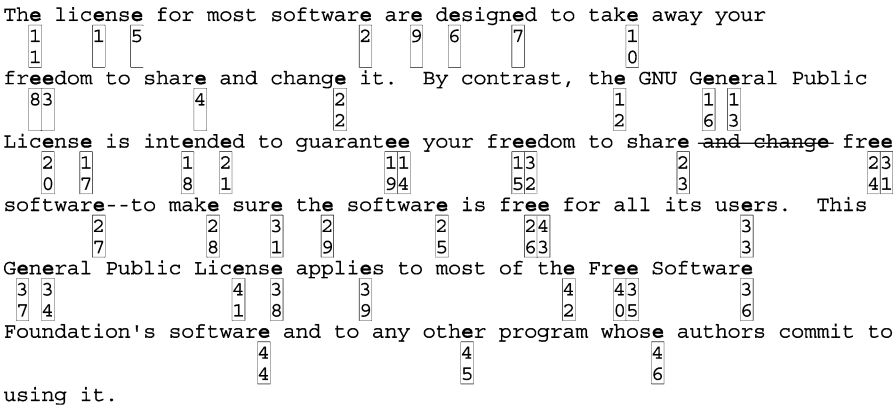


Fig. 6. Tampered document.

We assume the modifications only appear at a small part of the document. This assumption is reasonable because most unauthorized modifications are aimed at changing a few critical words rather than the whole document. We use the RSE decoding algorithm to determine the permutation length. Here we find the permutation length is 11 from the first, second, and the fourth lines. Since most of the document has not been modified, the number 11 is credible, and modification must be in line 3. Substituting line 3 with the permutation obtained from line 1, 2 and 4, the proper sequence for line 3 should be $\langle 33, 23, 27, 24, 31, 28, 29, 32, 30, 25, 26\rangle$. Thus, we detect the modification as missing a permutation target "e" between the 23rd and 24th letter "e"s.

The addition of permutation targets can be similarly detected. In cases where more accurate tamper detection is needed, we may increase the number of permutation targets, e.g., permuting each vowel respectively, or permuting characters together with words or even sentences. However, RSE tamper detection can only detect modifications to permutation targets. It is therefor advisable to choose most sensitive words or characters as permutation targets.

The tamper detection method is especially handy if used together with the RSE authentication method. This is because the length of the permutations used in the RSE authentication method is usually very short. For most of the cases, only a small part of all available permutation targets are permuted. It allows us to encode the same permutation multiple rounds by taking advantage of the remaining permutation targets, such that the RSE decoding algorithm can be more accurate and the tampered location is more easily identified.

III. Document Authentication With RSE

We have described RSE in the last section as a method to embed hidden data into electronic documents. RSE provides enough information carrying capacity to embed rights description for the document as well as other auxiliary data. In this section, we shall focus on issues relating to use of RSE for efficient document authentication. We would like the authentication process to have both merits of digital watermarking and digital signatures. RSE is a type of digital watermarking scheme. Now here the key point is to look for an authentication algorithm which has similar properties as that of a digital signature. That is, we can publish certain parameters of the algorithm for verification, but by using the document and the published parameters, one should not be able to forge the authentication. Let us look at the digital signature again. There are mainly three factors governing the authentication process: document digest, private key and public key. Here, the private key is used to link the authorized signatory with the document, while the document digest is the quantitative measure of the content. In this section, we shall describe a RSE authentication method, which is based on XTSP. Three key parameters used here are: the document digest, the permutation of RSE coding, and the cost matrix of the tour which is published in the trusted database. All three parameters are tied together using the computationally hard XTSP problem: the permutation defines the RSE encoding and the tour, with published cost matrix, and the total cost of the tour is equal to the context digest. Because of the complexity of message digest code and XTSP, it is not likely that one can forge a context digest with known permutation, or forge a permutation with known context digest.

A. Mathematical Background

Since 1989, there have been several attempts [19]–[25] to build cryptosystems based on NP-complete problems which use operations over small numbers or even bits. Most of these schemes have been proposed on the zero-knowledge interactive proof background without touching the authentication requirements. Here we propose an authentication method based on the XTSP [24], [25]:

Definition 3.1: Let G define a weighted graph (V, E) where V is a set of vertices, and E is a set of edges between members of V. $c_{i,j}$ denotes the cost for each edge from V_i to V_j. The XTSP problem is to find a tour T which visits each vertex once and only once in G, with the total cost $\sum c_{i,j}$ equals to a given cost L.

The NP-Completeness and hardness of XTSP has been proved in [24]. So far, there are no effective solutions for XTSP other than exhaustive search. For an n-city XTSP, the solution space contains $(n-1)!$ different tours. With $n = 41$, the complexity of XTSP is comparable to 160-bit key since $40! \approx 2^{160}$.

In the RSE authentication method, we use a variation of XTSP—Modular XTSP:

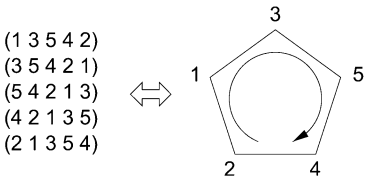


Fig. 7. Permutations and corresponding Hamiltonian cycle.

Definition 3.2: The modular XTSP is to find a Hamiltonian tour T in graph G such that $\mathrm{Cost}_C(T) = L(\mathrm{mod} 2^l)$, wherein l is a parameter which determines the security of modular XTSP.

For a graph of n vertices, there are altogether $(n-1)!$ different Hamiltonian tours, so T can be described using $\lceil \log_2(n-1)! \rceil$ bits. It is natural to consider both the length of $c_{i,j}$ and l equal to $\lceil \log_2(n-1)! \rceil$ so that the modular XTSP can yield $(n-1)!$ uniformly distributed Ls. In fact, $l = \lceil \log_2(n-1)! \rceil$ defines the most secure case of modular XTSP [24].

B. RSE Authentication Method

So far, we have established RSE coding and decoding schemes for embedding the authentication information, and have shown that modeling the permutation identification in RSE using exact travel salesman problem is an ideal scheme for authentication. Now in this section, we will establish the RSE authentication method for electronic documents. In an electronic document of n permutation targets, and for total of $(n-1)!$ permutations, we treat a very small subset of the permutations as valid, then authenticate the document by examining the presence of the permutation.

Given an n-permutation, we can create a directed Hamiltonian cycle by concatenating the first and last elements. And given a directed Hamiltonian cycle of length n, we can create n corresponding permutations, by regarding each node in the cycle as the starting point, as shown in Fig. 7. This relationship between permutation and Hamiltonian cycle enables us to authenticate the permutation by authenticating its corresponding Hamiltonian cycle.

For efficiency consideration, RSE authentication method authenticates a batch of documents together. These documents can have the same contents but created for different recipients or just have different contents. In business and administrative environment, the need for differentiating recipients or preparing a series of documents for a single transaction is very frequent, so our method is adaptable. Suppose there are N documents to be authenticated, we execute the following procedures (as shown in Fig. 8).

1) Choose a number n such that $n \geq 41$ and $n \times (n-1) > N$.
2) For each document, assign a distinct n-permutation $P_i, i = 1 \ldots N$. Without loss of generality, we require $P = (1, \ldots)$ (normal notation, so there are $(n-1)!$ different Ps). Generation of permutations can be done by using a pseudorandom number generator (PRNG) to generate some random numbers, convert the random number to $(n-1)$-permutation using RSE encoding algorithm, then insert 1 as the first element and adjust the following elements accordingly.
3) Generate Hamiltonian tour T_i from $P_i, i = 1 \ldots N$.

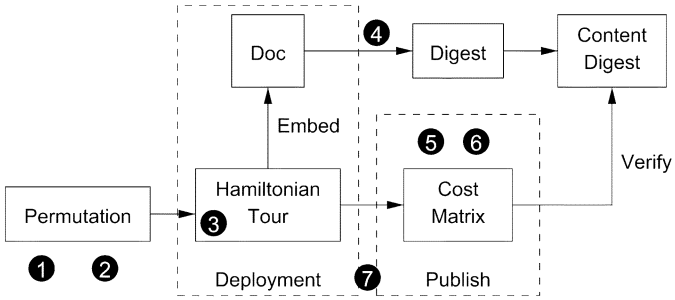


Fig. 8. RSE authentication flowchart.

4) For each document, extract all text content or certain critical text content, then create a $l = \lceil \log_2(n-1)! \rceil$ bits message digest of the contents using a collision-resistant one-way hash function $H(\cdot)$. The digests are denoted using $L_i, i = 1 \ldots N$.
5) Create an all-zero $n \times n$ cost matrix C, then solve the equation

$$\mathrm{Cost}_C(T_i) = L_i (\mathrm{mod} 2^l), \quad i = 1 \cdots N \tag{3}$$

by adjusting $c_{i,j}$ in C. Since there are $n \times (n-1)$ unknowns in matrix C ($c_{i,i} = 0$ for $i = 1 \cdots N$) and (3) only contains $N < n \times (n-1)$ constraints, C is always solvable using linear algebra method.
6) Assign all unused $c_{i,j}$ random values. Then calculate

$$c_{i,j} = c_{i,j} (\mathrm{mod} 2^l), \quad i, j = 1 \cdots N \tag{4}$$

With this step, $c_{i,j}$ has been limited to l bits.
7) Finally, publish the cost matrix C as the verification key, and embed P_i into corresponding document using the RSE watermarking scheme.

For verification of the document, the verifier first calculates the message digest L_i' from the document using the same one-way function $H(\cdots)$. The verifier then extracts the permutation P_i from the document, converts it to Hamiltonian tour T_i, and verifies if

$$\mathrm{Cost}_C(T_i) = L_i' \ (\mathrm{mod} 2^l), \quad l = \lceil \log_2(n-1)! \rceil.$$

The selection of one-way hash function $H(\cdots)$ needs special consideration. It must be able to output $\lceil \log_2(n-1)! \rceil$ bits message digest. There have no such variable length one-way functions been proposed except for the HAVAL [26] (outputs 128, 160, 192, 224, 256 bits) and SHA-V (outputs 128, 160, 192, 224, 256, 288, 320 bits) algorithms. While truncating hash values to a lower number of bits is possible, concatenating shorter values to form a longer value reduces security. We recommend selecting proper n values so that $\lceil \log_2(n-1)! \rceil$ are comparable to hash function outputs. Recommended values are $n = 41, (\log_2(40!) \approx 160), n = 47, (\log_2(46!) \approx 192), n = 52, (\log_2(51!) \approx 224), n = 58, (\log_2(57!) \approx 256), n = 63, (\log_2(62!) \approx 288), n = 68, (\log_2(67!) \approx 320)$. For number of documents more than $68 \times (68-1) = 4556$, it is possible to partition documents into several groups, then generate cost matrix C for each group.

Once a cost matrix C has been fixed, adding more documents for authentication requires re-calculating the whole C. This is because an attacker can otherwise compare the cost matrix before and after adding new documents to determine the newly used edges. For verification, it means the verifier must always keep his copy of matrix C up-to-date. This resembles the verification of digital signature where the verifier must retrieve the signer's public information from a trusted server. Nevertheless, if the cost matrix C has been fully utilized (authenticate $n \times (n-1)$ documents), the verifier only needs to retrieve an average l-bit value for one document.

C. Security Analysis

The security of the authentication scheme is easily verified.

- For the total $(n-1)!$ possible Hamiltonian tours, we treats N of them to be valid. So the possibility that a random permutation be erroneously considered valid is

$$\frac{N}{(n-1)!} < \frac{n \times (n-1)}{(n-1)!}$$

  For $n = 41$, this figure is about 2×10^{-45}, which means such a coincidence is really rare. We do not specifically require distinct Hamiltonian tours produce different costs. It is very unlikely because the space for cost values is at least as large as the space for Hamiltonian tours (2^l vs. $(n-1)!$). There is no way to prevent collisions except enumerating all tours, which is an astronomical figure.
- In order for a deliberate attacker to forge a document which can pass the verification process, he must be able to do one of the following things.
  1) He creates a new document and generates a hash value L'. In order to embed a correct n-permutation P' into the document, he must solve the modular XTSP to find a Hamiltonian tour T' that satisfies $\mathrm{Cost}_C(T') = L' \pmod{2^l}$. The mathematical background shows it is intractable.
  2) He selects a Hamiltonian tour T' and calculates $L' = \mathrm{Cost}_C(T') \pmod{2^l}$. Now he must reverse the one-way hash function $H(\cdots)$ in order to create a meaningful document and relevant information that hashes to L'. It is also intractable since the one-way hash function is collision resistant.
- In RSE authentication method, the verification key is the cost matrix C, the authenticator is the permutation P or Hamiltonian tour T, and the embedding of $P/(T)$ into electronic document is through the RSE watermarking scheme. It can be figured out that our method does not have a specific authentication key, which means the sender Alice has nothing secret. What she has is the original document and authority to publish C. If Alice puts C onto a trusted server, then the attacker must have had to compromise the servers in order to conduct a successful attack.

In conclusion, RSE authentication method relies on the hardness of the algorithms and the proper management of the document signing process. We have shown that the RSE authentication algorithm is based on XTSP, and the complexity of the algorithm is of the order of $(n-1)!$. The document signing process is very easy to manage when compared to PKI.

IV. Conclusion

In this paper, we started by reviewing two types of electronic document authentication methods: digital signature and digital watermarking. Digital watermarking is content-based while digital signature is key-based and very secure. We then proposed the RSE authentication method which combines the merits of both.

The major difference between the RSE watermarking scheme and existing watermarking schemes is that our scheme takes advantage of the redundancies in the page description languages of electronic documents. The redundancies are captured as render sequences. By manipulating render sequences, we achieve information carrying capacity that is several orders of magnitude larger than all existing schemes. The RSE watermark scheme is robust in terms of surviving file format transcoding. This feature achieves interoperability by bridging rights description and authenticator from one document format to another.

Based on the RSE watermarking scheme, RSE authentication method adopts modular XTSP to authenticate the document. The security of RSE authentication method is guaranteed by the intractability of XTSP. The advantage of RSE authentication method over digital signature is its small authenticator size. With this feature RSE authentication is adaptable to very short documents. Another feature of RSE authentication is its compatibility with most popular document formats. It can be integrated into existing systems with only minor changes at the creation and rendering ends of the whole workflow. RSE authentication method facilitates the "management of rights holders' relationship" by establishing trust among parties involved in document exchange. It can thus be a major building block in the whole DRM system for electronic documents.

References

[1] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL: CRC, 1997.

[2] I. J. Cox and M. L. Miller, "Electronic watermarking: The first 50 years," in *Proc. IEEE Int. Workshop on MultiMedia Signal Processing*, Antwerp, Belgium, Sep. 2001.

[3] J. Brassil, S. H. Low, N. F. Maxemchuk, and L. O'Gorman, "Marking text features of document images to deter illicit dissemination," in *Proc. 12th IAPR Int. Conf. Pattern Recognition, Computer Vision & Image Processing*, Oct. 1994, vol. 2, pp. 315–319.

[4] ——, "Electronic marking and identification techniques to discourage document copying," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 8, pp. 1495–1504, Aug. 1995.

[5] J. Brassil, S. Low, N. F. Maxemchuk, and L. O'Gorman, "Hiding information in documents images," in *Proc. Conf. Information Sciences and Systems (CISS-95)*, 1995.

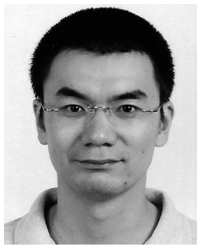
[6] S. H. Low, N. F. Maxemchuk, J. Brassil, and L. O'Gorman, "Document marking and identification using both line and word shifting," in *Proc. INFOCOM*, Boston, MA, Apr. 1995, pp. 853–860.

[7] J. Brassil, "September—Secure electronic publishing trial (poster)," in *Proc. 1st ACM Int. Conf. Digital Libraries*, Bethesda, MA, Mar. 20–23, 1996, p. 177.

[8] S. H. Low and N. F. Maxemchuk, "Performance comparison of two text marking methods," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 4, pp. 561–572, May 1998.

[9] J. Brassil, S. Low, and N. F. Maxemchuk, "Copyright protection for electronic distribution of text documents," *Proc. IEEE*, vol. 87, no. 7, pp. 1181–1196, Jul. 1999.

[10] N. Maxemchuk and S. Low, "Marking text documents," in *Proc. Int. Conf, Image Processing (ICIP-97)*, Washington, DC, Oct. 1997, pp. 13–16.
[11] S. H. Low, N. F. Maxemchuk, and A. P. Lapone, "Document identification for copyright protection using centroid detection," *IEEE Trans. Commun.*, vol. 46, no. 3, pp. 372–383, Mar. 1998.
[12] N. Memon and P. L. Vora, "Authentication techniques for multimedia content," in *Proc. SPIE*, 1999, vol. 3528, pp. 412–122.
[13] *Adobe System Incorporated* (in Postscript language reference), , Feb. 1999 [Online]. Available: http://partners.adobe.com/public/developer/en/ps/PLRM.pdf, third edition, pp. 176–177
[14] S. Johnson, "Generation of permutations by adjacent transpositions," *Mathem. Comput.*, vol. 17, pp. 282–285, 1963.
[15] H. Trotter, "Algorithm 115," *Commun. ACM*, vol. 5, pp. 434–435, 1962.
[16] M. Gardner, "Mathematical games," *Scientif. Amer.*, pp. 122–125, Nov. 1974.
[17] J. M. Hall, in *Proc. Symp. Pure Mathematics*, Providence, RI, 1963, vol. 6, p. 203.
[18] R. A. Brualdi, *Introductory Combinatorics*. Upper Saddle River, NJ: Prentice-Hall, 1999.
[19] A. Shamir, "An efficient identification scheme based on permuted kernels," in *CRYPTO89*, 1990, pp. 606–609, Lecture Notes in Computer Science No. 435.
[20] J. Patarin and P. Chauvaud, "Improved algorithms for the permuted kernel problem," in *Proc. CRYPTO93*, D. R. Stinson, Ed., 1994, pp. 391–402, Lecture Notes in Computer Science No. 773.
[21] J. Stern, "A new identification scheme based on syndrome decoding," in *Advances in Cryptology—Proc. CRYPTO'93*, D. R. Stinson, Ed., 1994, pp. 13–21, Lecture Notes in Computer Science No. 773, Springer.
[22] D. Pointcheval, "A new identification scheme based on the perceptrons problem," *Lecture Notes in Computer Science*, vol. 921, p. 319, 1995.
[23] J. Stern, "Desiging identification schemes with keys of short size," in *Advances in Cryptology, Proc. CRYPTO'94*, 1995, pp. 164–173, Lecture Notes in Computer Science No. 839 Springer.
[24] S. Lucks, "How to exploit the intractability of exact TSP for cryptography," in *Fast Software Encrypt.*, 1994, pp. 298–304.
[25] ——, "How traveling salespersons prove their identity," in *IMA: IMA Conf. Cryptography and Coding, LNCS*, 1995.
[26] Y. Zheng, J. Pieprzyk, and J. Seberry, "Haval—a one-way hashing algorithm with variable length of output," in *ASIACRYPT*, 1992, pp. 83–104.



Baoshi Zhu received the B.Eng. and M.S. degrees from Shanghai Jiaotong University, Beijing, China, and the Ph.D. degree in computer science from the National University of Singapore.

He then joined Trustcopy Pte. Ltd., Singapore, a high-tech spin-off from the Kent Ridge Digital Labs, where he is the Innovation Director. His research interests are Digital Rights Management, paper/electronic document protection, and secure printing.

Jiankang Wu received the B.Sc. degree from the University of Science and Technology of China and the Ph.D. degree from Tokyo University, Tokyo, Japan.

He is currently a Professor in the Graduate School, Chinese Academy of Sciences, Beijing, China, and Director of the Sensor Networks and Application Research Center. He was Principal Scientist at the Institute for Infocomm Research, Singapore, leading new research initiatives in the area of physioinformatics and embedded sensor network systems. Prior to joining the Institute for Infocomm Research (known as Kent Ridge Digital Labs in 1998–2001 and the Institute of Systems Science before 1998) in 1992, he was a Full Professor at the University of Science and Technology of China. He also worked in universities in the U.S., U.K., Germany, France, and Japan. He pioneered several researches in the area of visual information processing, including adaptive image coding in late 1970s, object-oriented GIS in early 1980s, face recognition systems in 1992, content-based multimedia indexing and retrieval in the 1990s, and rights management for multimedia contents in late 1990s. His research led to two spin-off companies.

Dr. Wu received nine distinguished awards from the nation and the Chinese Academy of Science.

Mohan S. Kankanhalli (M'92) received the B.Tech degree in electrical engineering from the Indian Institute of Technology, Kharagpur, and the M.S./Ph.D. degree in computer and systems engineering from the Rensselaer Polytechnic Institute, Troy, NY.

He then joined the Institute of Systems Science in Singapore, where he mainly worked on content-based multimedia information retrieval. During 1997–1998, he was with the Department of Electrical Engineering, Indian Institute of Science, Bangalore. He then moved to the School of Computing at the National University of Singapore, where he is a Professor. His current research interests are in Multimedia Systems (content processing, retrieval) and Multimedia Security (surveillance, watermarking, and authentication).

Dr. Kankanhalli is on the editorial boards of several journals, including the IEEE TRANSACTIONS ON MULTIMEDIA, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, the *ACM Transactions on Multimedia Computing, Communications, and Applications*, and the *ACM/Springer Multimedia Systems Journal*.