Techniques for Content-Based Graph Authentication

Heather Yu and Xiangyang Kong
Panasonic Information and Networking Technology Lab

Wayne Wolf
Princeton University

In addition to images and text, more and more documents use graphs for system and idea illustration. Compared to gray-scale images, graphs are more difficult to watermark because of their binary nature. We propose two methodologies for digital graph authentication, at the object and pixel levels, that detect and localize alterations.

For as long as humans have communicated with one another, we've had concerns about maintaining confidentiality. Recent advances in communications, networking, and multimedia information access technologies have made authentication of digital multimedia content a more important research area. The objective is to authenticate information and protect it from unauthorized alteration.

Researchers have proposed various schemes using conventional cryptographic methods or digital watermarking for protecting images,[1-3] plain text,[4-7] audio,[7,8] and video.[9,10] However, little has been done to protect graphs—an important part of information in documents. This article focuses on text documents that we can print on paper. Specifically, we present two new schemes for authenticating graphs in these text documents. Roughly speaking, a document can consist of plain text, images, and graphs. We define a graph as a diagram symbolizing a system or illustrating a statement. In this article, we refer to only those digital graphs with binary pixel values (1 bit per pixel), but our methodology applies to graphs with multiple bits per pixel.

The problems we need to solve include first designing a simple scheme for authenticating graphs in text documents that are suitable for common document editors without requiring additional coding. Second, we need to reliably detect alterations after file-format changes and/or photocopying. Third, if we adopt watermarking techniques, how can we authenticate a 1 bit/pixel graph without noticeably altering the visual content and still easily localize any meaningful (or say damaging) alterations?

To address these issues, we propose an intelligent symbolic representation language for graph authentication that lets us use suitable text document authentication algorithms to authenticate the graph together with the text in context. We propose dual-layer and coalescing authentication using visible and invisible pixel-level authentication together with object-level authentication schemes.[11] This provides protection capabilities even after file-format changes.

Authentication scenarios

Let's define I as the original document that the owner $O1$ or owners $O1$, $O2$, …, On will authenticate in the case of a contract. The authenticated copy of document I is $\tilde{I}$. We define G and $\tilde{G}$ as the original and the authenticated copy of a graph, respectively. We also denote R as an authorized receiver, whereas A is an attacker, or unauthorized receiver. The following scenarios illustrate our work's application value and objectives.

- $I_1 \in O1$. $O1$ wants to check if her document I_1 is authentic. The document's content is genuine, especially the sensitive information, such as $1,000 or by 01 June 1999.

- $I_1 \in O1$. $O1$ might need to send I_1 to R and might wish to grant R read permission but not write permission. Or, $O1$ might want to prevent any alterations and localize the alteration made by an attacker A who gets I_1 from $O1$ and then sends it to R. Or, $O1$ might want everyone to be able to read I_1 but only let R and herself modify the document.

- $I_1 \in O1 \cap O2$. I_1 might be a contract between $O1$ and $O2$. If the copy in $O1$'s hand differs from the one in $O2$'s, $O1$ might want to prove that $O2$'s copy is a tampered copy of the original contract by checking the authenticity of $O2$'s copy. In addition, $O1$ might want to point out exactly where $O2$ altered the original contract.

In this article, we concentrate on the $I_1 = G_1$ scenario unless otherwise specified.

1070-986X/01/$10.00 © 2001 IEEE

New capabilities

Researchers have proposed document authentication techniques to ensure the integrity of a wide variety of electronic documents, such as presentations, contracts, military orders, expense reports, and proprietary documents. The best example is the Adobe Acrobat format. Adobe adopted a digital signature capability in its 4.0 version. It allows either the use of a certification authority or self-trusted authentication. Once a document has been converted to a PDF, Acrobat supports both key-based and biometric digital signatures, with which users can authenticate authorship, track alterations, and verify approvals of documents. While this added authentication capability provided by Acrobat is useful and appealing, the drawback is its inability to support authorized modification and file-format changes. Unlike Microsoft Word or most of the other commonly used document editors, Acrobat doesn't support easy modification. That is, once a document is generated in PDF format, users can't modify it arbitrarily, which introduces some limitations. In addition, because the digital signature is attached to the document, it's lost once users change the document format or print it out.

To support document identification even from a photocopy, researchers have proposed digital watermarking technologies and have continuously studied image watermarking for several years. Text and graphs are often referred to as binary images. This binary nature makes it particularly hard to insert any watermark due to the low capacity of perceptual invisible noise. In other words, a minimal alteration of bits in a binary graph can result in a substantial change in the graph's appearance and content.

Previous methodologies for text authentication mainly rely on altering the word or line spacing or the length of strokes. These methodologies might be useful for certain applications of plain-text protection, but we can't extend them to graph authentication even though they share the same binary nature. Graphs often don't observe the same characteristics as text, even at the pixel level. For instance, in a flow diagram, each node's shape might be important whereas it often has fewer characters compared to a text paragraph. The number of objects that observe a vertical serif can be as low as several percent of the total number of objects. Here, we refer to an object as a line, character, or curve. Other kinds of graphs might not observe line spacing or vertical serifs (see Figure 1).

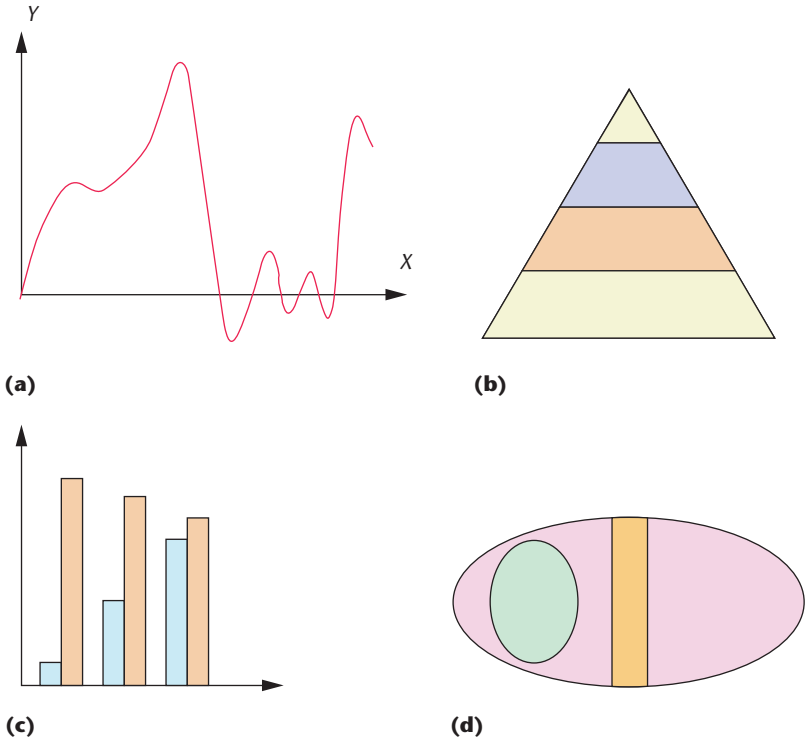


Figure 1. Example graphs that don't observe line spacing or vertical serifs.

On the other hand, a graph's critical information is often in the object level rather than the pixel level. For instance, a useful application for document copy and copyright protection is to provide different levels of access rights to different users while detecting and localizing alterations. In a graph, such as the procedure in Figure 2, the important information is in the annotations attached to each node and the connections between nodes that illustrate their relationships. Whether each box is a different size, lines are different lengths, or a node is tilted to the left or right generally isn't that important. Consequently, the desired target of authentication is at the flow level instead of the pixel level. (It's worth pointing out here that this observation is generally true for plain text as well—namely, the sensitive information is contained in the characters rather than the pixels. That is, the text's font or size seldom alters its content. Hence, in most cases, the desired target for text authentication is at the character level.) In other words, we can define characters as the objects of content, or physical entities, for marking and protecting text documents.

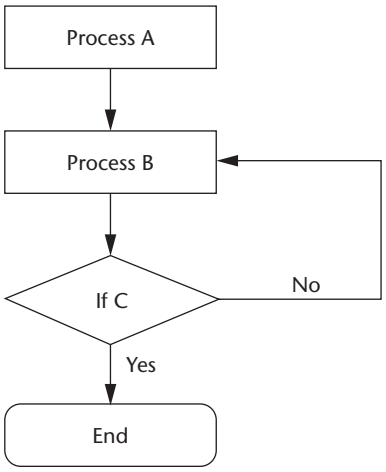


Figure 2. A simple flow diagram.

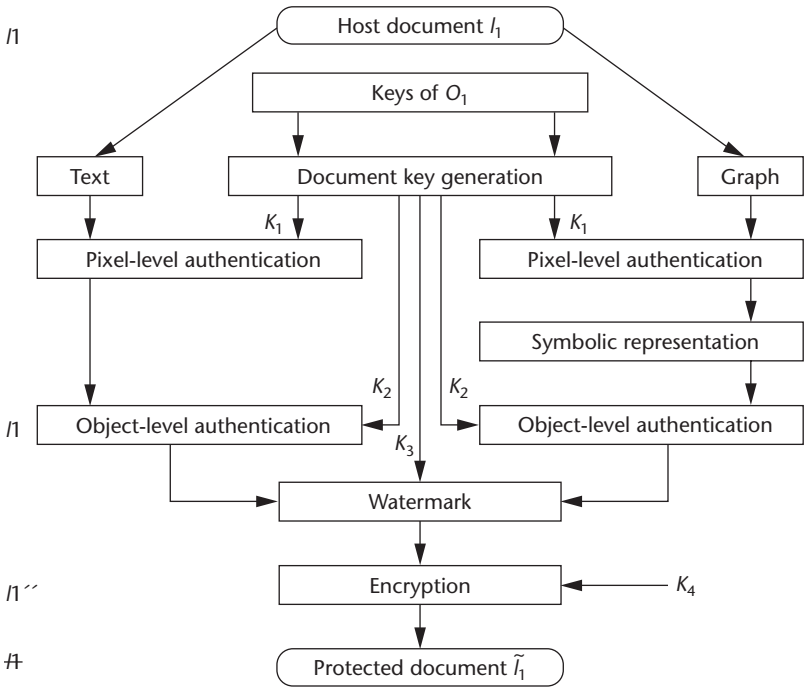


Figure 3. A general framework for one-party owned document authentication.

Sometimes it's desirable for commonly used document editors to easily adopt and support an authentication scheme without special coding. A digital signature with the ability to survive document format changes is also an appealing function.

General framework

Obviously, if we simply copy the previous proposed text authentication scheme, such as a stroke-length-based scheme, or modify it to a corresponding scheme in graph, such as a line-length-based scheme, it won't fulfill our authentication goals. We need new techniques for authenticating graphs in text documents.

This article presents new mechanisms that can authenticate binary graphs in text documents. By building a bridge from graph to text at the character level, we can authenticate graphs using a suitable text document authentication algorithm. When we require pixel-level precision of a graph, we can add a pixel-level authentication. This layer lets the owner detect and localize a graph change at the pixel level.

Figure 3 illustrates the general framework of a one-party owned system. The hierarchical layout yields the application of several aforementioned scenarios. The first hierarchy is the pixel-level authentication followed by an object-level authentication hierarchy. We do these with owner *O*1's private key. Notice that either the pixel-level or object-level protection is optional, depending on the application. For ultimate protection, however, we recommend a dual-layer protection with a pixel-protection layer plus an object-protection layer since the two layers are orthogonal. Next, users can insert a meaningful watermark, such as a company logo, if desirable. At last, we can encrypt the authenticated documents, text plus graph, with a public-key encryption algorithm for secure transmission. We can add the watermarking layer either before or after the authentication layer. This again, depends on different applications. We can then grant access authorization by distributing different keys to different users. For example, in the read-only case, *R* will be given the public decryption key K_4 only. In the case of a multiparty owned document authentication, each party has a private key, the authentication is done by generating a key set with the private key from every party. Any attempt to modify the document without a key will be unsuccessful.[12]

Furthermore, we can use a context-dependent one-way hash in the system, as Wu et al. suggested.[3] This provides an addition gain in robustness compared to earlier approaches.

Symbolic graph representation

Text and graphs, although sharing the same binary nature, have different object-level representations. Graphs often contain lines and curves in addition to text characters, so we can't directly authenticate them with the same algorithm we use for text authentication. However, if we can create a bridge from the graph to the text, such as a symbolic representation of the graph, then we can realize graph authentication with proper text authentication algorithms and therefore authenticate graphs along with the text in context.

When a graph's sensitivity is on the object level, we can represent it precisely with a series of relationship and specification symbols that specify the nodes' exact characteristics and relationships, along with the node annotation. Let's use Figure 2, a typical system procedure diagram, to illustrate the idea. The figure consists of nodes and lines. For illustration purposes, we simplified the annotation of each node. We can represent this diagram with a series of relationship symbols along with the node annotations:[13]

"<N_1{'Process A', #1, &reg, @mid} → N_2{'Process B', #1, &reg, @mid} → N_3{'If C', #3, &reg, @mid} → <N_4{'End', #2, &reg, @mid}|yes; N_2|no>> Fig2 A simple flow diagram."

In this symbolic representation, N_1, N_2, and so on are node names with the property of each node contained in {…}, < > is a tuple, and → and | are relationship symbols (see Table 1). We can describe the properties of nodes and lines—the shape, size, color, and position—with the specification symbols. For those specification insensitive graphs, we can simply ignore the symbols between each pair of { }. In specification sensitive graphs, the specification symbols in each pair of { } give different detail levels. This hierarchical representation provides additional flexibility.

After symbolization, we transfer a multimedia document (with text and graphs, see Figure 4) to a symbolized document (text only, see Figure 4b) that we can authenticate with a suitable plain-text authentication algorithm.

For example, we can use a 2D or multidimensional checksum to verify the authenticity, or more elegantly, choose a conventional authentication algorithm such as the Secure Hash Algorithm (SHA). Here are two simple examples. (This is merely an illustration. The optimum choice of authentication algorithms for real-world application scenarios is out of the scope of this article. Interested readers can refer to cryptography literature.)

Let $T(p, q)$ represent the (p, q)th character. $S(p, q) = s^1(p, q)\ s^2(p, q)\ \ldots\ s^J(p, q) = f(T(p, q))$ is a coded representation of $T(p, q)$ via map f. $s^1(p, q)\ s^2(p, q)\ \ldots\ s^J(p, q)$ represent the first, the second, and so on to the Jth bit of $S(p, q)$ that are in the order of the most to the least significant bit. Let

$$\mathrm{Sum}_p^j = \sum_{p=1}^{P} s^j(p,q)$$

and

$$\mathrm{Sum}_q^j = \sum_{q=1}^{Q} s^j(p,q)$$

where P and Q are dimensional sizes. Thus, we can localize the position (p, q) of any alteration $\mathrm{Sum}_p' \neq \mathrm{Sum}_p$, $\mathrm{Sum}_q' \neq \mathrm{Sum}_q$.

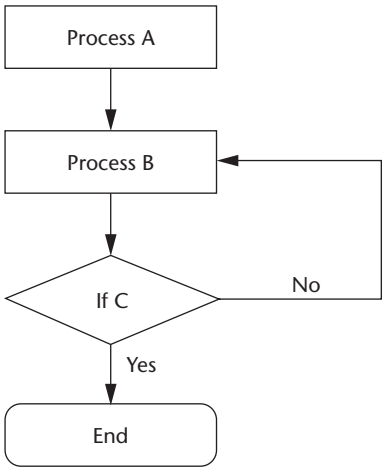
Of course, using a content-dependent one-way hash function gives a higher level of security. We can use the methodology suggested in Wu et al.[3] Let B denote the block size and K denote a private key. In the case of a multiparty document, K is a function of $K_{O1}, K_{O2}, \ldots$. That is, $K = f1(K_{O1}, K_{O2}, \ldots)$. Assume it has J bits of code. The first to $(J - 1)$th bit are the code bits, and the lowest bit, Jth, is the verification bit. Figure 5 plots the object values of the symbolized text paragraph I in Figure 4. We named it the *teximage*, the symbolized text stream's gray-scale image that is a mapping of each object value to a gray-scale pixel value. It uses 9 bits of code. We chose the one-way hash algorithm MD5. The encoding procedure is as follows:

1. Pad the source text I an exact multiple of 512 in length.

Table 1. Sample Symbols for a Graph's Symbolic Representation.

Relationship Symbols	Definition
< >	Tuple
∩	And
∪	Or
≠	Not
→	Partent → child
↔	Sibling relation
⇔	Twin relation
←	Child ← parent
>	Contain relation
|	Condition
.	Unconnected

Specification Symbols	Definition
&	Size
#	Shape
@	Position
©	Color



(a)

"<N_1{'Process A', #1, &reg, @mid} → N_2{'Process B', #1, &reg, @mid} → N_3{'If C', #3, &reg, @mid} → <N_4{'End', #2, &reg, @mid}|yes; N_2|no>> Fig2 A simple flow diagram."

(b)

Figure 4. (a) A sample document paragraph and (b) its corresponding symbolized verson.

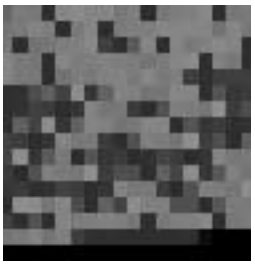


Figure 5. The teximage corresponding to Figure 4b.

2. For each 128-length set, I_o, choose its neighborhood set, $\underline{I}_o$ = 512 characters with $I_o \subset \underline{I}_o$.

3. Assume $S_o = \{S_o(i),\ i \in [1,128]\} = \{s^1{}_o(i)\ s^2{}_o(i) \ldots s^l{}_o(i)\} = f(I_o)$ and $\underline{S}_o = \{\underline{S}_o(i),\ i \in [1,512]\} = \{\underline{s}^1{}_o(i)\ \underline{s}^2{}_o(i) \ldots \underline{s}^l{}_o(i)\} = f(\underline{I}_o)$ are coded representations of I_o and $\underline{I}_o$ respectively.

4. Concatenate the code bits of the neighborhood set $\underline{I}_o$.

5. Calculate the 128-bit hash value of it, $h_o = H(\underline{S}_o)$.

6. Generate message h_o'=Sgn(K, h_o) by signing h_o with a public cryptography method.

7. Put h_o' into the Jth bit, the lowest bit, of $S_o(i)$—that is, let $s^l{}_o(i) = h_o'(i)$, $i \in [1,128]$.

The decoding process resembles the encoding process with the verification done through an XOR operation:

$$Auth_o(i) = \tilde{h}_o'(i) \oplus s^J{}_o(i)$$

If $Auth_o(i) = 1$ for ®$i \in [1,128]$, the I_o set has been altered.

When we visibly attach a teximage of an authentication value or a digital signature to a document, we can anticipate its survivability over file-format changes and digital-to-analog and analog-to-digital (DA-AD) transformations.

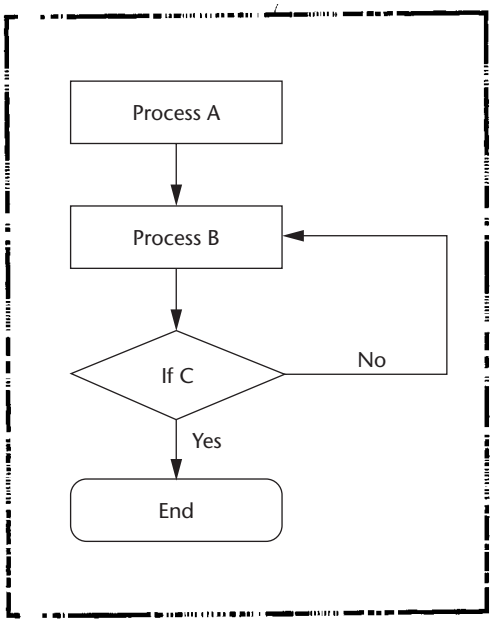


Figure 6. An authenticated graph using a bounding box.

Pixel-level authentication

For a position sensitive or pixel-level precision critical graph, object-level algorithms can't fulfill the goal. We present two schemes for pixel-level graph authentication. Method 1 leaves a visible mark on the graph, but method 2 doesn't.

For method 1, let XxY = 128 be the defined block size. Cut G into XxY blocks. Assume the number of blocks is L.

1. We concatenate the bits of the (x, y)th pixel of every block to the first block and form an L-bit truncated image TrunG. Therefore, we generate an L bits/pixel image TrunG, with image size XxY, of graph G. Let TrunG$(x, y)^l$ denote the lth bit of pixel (x, y) of TrunG. Note it's desirable to form the truncated image TrunG so that TrunG$(x, y) \neq 0$. Also note that to get a higher level of protection we should use a random number generator to cut the graph.

2. Collect all the bits of all XxY pixels into a $XxYxL$-bit message $M1$. Pad $M1$ into an exact multiple of 512 in length with as many 0s as needed to get message $M1'$.

3. Compute the 128-bit hash value of it using MD5: $M2 = h(i) = H(M1')$.

4. Sign $M2$ with a public-key cryptography method and generate $M3 = h'(i)$ = Sgn(K, $M2$).

5. Generate a bounding box Gb of G with $Gb(i) = h'(i)$. (Figure 6 illustrates this idea.)

This scheme is especially suitable for pixel-level graph authentication because an added bounding box won't interfere with the graph's appearance. (Note that in the case of text, an added bounding box isn't acceptable because of the change in appearance.) In addition, by enlarging the width of the box and repeating the 128 bits multiple times along the bounding box, the box can easily survive distortion caused by common signal processing, even photocopying or faxing.

An alternative method uses a 1D or 2D bar code instead of a bounding box for authentication (see Figure 7).

When invisible authentication is necessary or desirable, we can use method 2, which is a less robust scheme that modifies the graph itself.

1. Perform step 1 from method 1.

2. Pick 1 bit TrunG$(x, y)^1 = 1$ out of the L bits of each pixel (x, y) in TrunG to be the verification bit. For better imperceptibility and a higher level of security, make sure the bits are as spread out as possible.

3. Collect the rest of the $(L - 1)$ bits from all XxY pixels into an $XxYx(L - 1)$ bit message $M1$. Pad $M1$ into an exact multiple of 512 in length with as many 0s as needed to get message $M1'$.

4. Compute the 128-bit hash value of it using MD5: $M2 = h(i) = H(M1')$.

5. Sign $M2$ with a public-key cryptography method and generate $M3 = h'(i) = \text{Sgn}(K, M2)$.

6. Embed $M3$ into TrunG:

- If $h'(i) = h'((y - 1)*X + x)) = 0$ and |TrunG(x, y)| is odd, let TrunG$(x, y)^1 = 0$.
- If $h'(i) = h'((y - 1)*X + x)) = 1$ and |TrunG(x, y)| is even, let TrunG$(x, y)^1 = 1$.

  Where |TrunG(x, y)| denotes the cardinality of TrunG(x, y)—that is, the number of bits that are 1s among the L bit of TrunG(x, y).

We can decode the graph in a similar way.

The advantage of the second scheme is the invisibility and the disadvantage lies in its low robustness. For both methods, however, we prefer a coalescing method, which we'll describe later. Instead of using the pixel-level value to generate the hash value, it uses the object-level (character) value of G to generate the hash value and embeds the hash value into the pixels. Figure 8 illustrates two sample results. We cropped Figures 8c and 8d from our sample diagram in Figure 2.

Techniques for content-based graph authentication

So far, we've used flow diagram examples to demonstrate how we can do content-based graph authentication at the object and pixel level. We can authenticate other graphs, such as those in Figure 1, using the same methodology. Note that different schemes have different robustness levels, and users should choose them based on specific application requirements. In some way, our schemes utilize each of the following techniques.

Dual-layer protection

Visible object-level authentication has a better survivability with regards to various kinds of processing noise. Even if the font, paragraph layout, or page layout changes, we might still verify the content's authenticity. That is, visible object-level authentication is more robust compared to invisible pixel-level authentication. However, an object-level algorithm is powerless when a graph requires pixel-level precision or transparency, yet new coding isn't applicable.

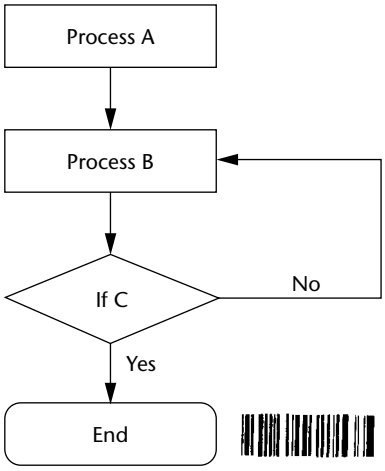


Figure 7. An authenticated graph using a barcode.

Because the proposed object-level signature is orthogonal to the pixel-level signature, a more plausible scheme is to authenticate the graph with a pixel layer followed by an object layer using invisible and visible marking.

Coalescing

Coalescing differs from dual-layer protection. To authenticate I with N symbols—for example, to authenticate the symbolized paragraph in Figure 4b, which has $N = 248$ characters—we

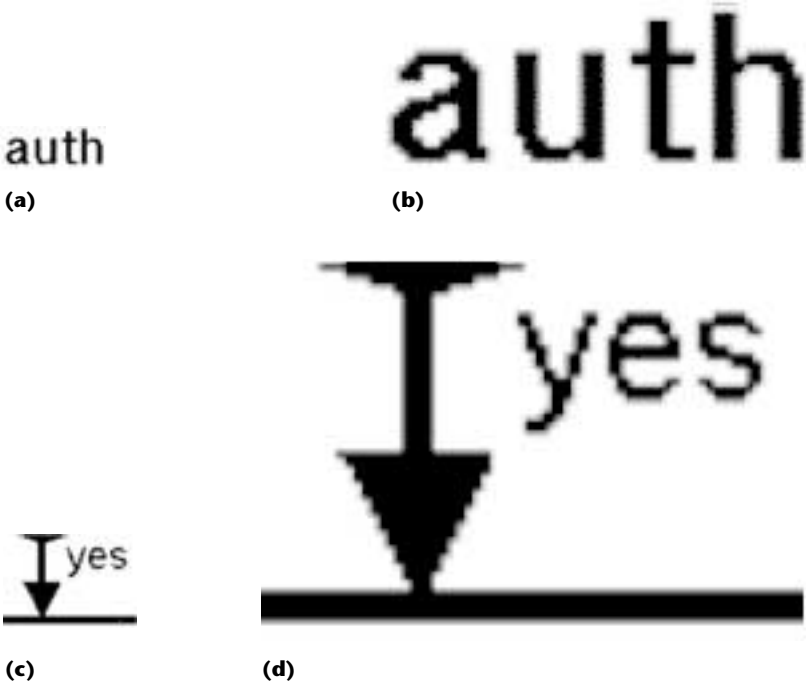


Figure 8. Sample results from the pixel-level authentication invisible marking method (method 2). To give a better view of the results, we enlarged each example (b and d) to approximately 400 percent of its original size (a and c).

Table 2. Comparison result for graph

Authentication Method	Without Content-Dependent One-Way Hash		With Content-Dependent One-Way Hash					
					Coalescing		Duel level with coalescing	
Capability	Traditional line spacing	Traditional serif length	Traditional line spacing	Traditional serif length	Method 1	Method 2	Method 1	Method 2
Special coding	Needed	Needed	Needed	Needed	Not needed	Needed	Depends	Needed
Imperceptibility	Can be good	Can be good	Can be good	Can be good	n/a	Can be good	n/a	Can be good
Detectability*	Bad	Bad	Bad unless enough lines	Bad unless enough serifs	Good	Okay	Good	Good
Pixel-level detectability	Bad	Bad	Bad unless enough lines	Bad unless enough serifs	Good	Okay	Good	Okay
Localization ability**	Bad	Bad	Bad unless enough lines	Bad unless enough serifs	Good	Okay	Good	Good
Print, copy, and fax	Bad	Bad	Bad	Bad	Good	Bad	Good	Bad
Robustness to scaling	Can be good	Can be good	Can be good	Can be good	Good	Can be okay	Good	Can be good

* Detectability measures the correct detection rate when a certain number of objects is altered in the document and the processing noise is otherwise zero. This is because of the trade-off between imperceptibility and robustness of those methods. In general, we can achieve better robustness with more perceptibility.

** Localization ability measures the ratio of localized alterations.

compute the one-way hash of I at the character level first. We do this by putting all the bits of the 248 characters together, padding it to an exact multiple of 512 in length, and calculating its hash value. Then, we sign the hash value and embed the signed hash value at the pixel level using either one or both invisible and visible marking.

Embedding watermarks

We can also insert a fragile watermark other than the authentication value, such as an identification number or company logo, into the graph for copyright protection. An orthogonal feature is desirable in many cases. Due to the low data hiding capacity $C(G)$ of binary graph, we can only do this if $C(G)$ is greater than $h(i)$ in length.

Color and gray-scale graphs

We can extend the algorithms we've proposed to authenticate multibit-per-pixel graphs. For the object-level algorithm, all it takes is an additional set of symbols. For the pixel-level algorithm, however, a simple way is to extend a one-bit representation to a multibit representation.

It's important to note that with invisible marking, when we extend from binary to multiple bits, the host graph's data-hiding capacity is subsequently increased. Consequently, the problem's complexity is relatively reduced.

A comparison study

We compared our graph-authentication algorithms to the space-shifting and serif-modification methods. Table 2 summarizes our comparison results. We mostly measured them on the object level and performed alterations on the character level for text and the subnode or line level for graphs, unless otherwise specified.

Notice that when we prepend the hash value to the document, we don't need special coding for visible object-level authentication; otherwise, we do. Similarly, in the case of pixel-level (or coalesced) authentication, we don't need special coding with method 1, but we do with method 2. Here, special coding means a new code other than the commonly accepted codes, such as ASCII code and Unicode.

Conclusion

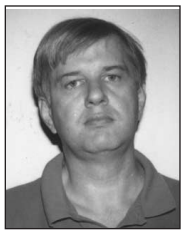
The techniques we proposed here are by no means a complete solution to all the problems in graph authentication. Many questions still remain. The sample testing results show each method's weakness and indicate the directions for future work. In some applications, a more robust, invisible marking system would be more desirable. Nevertheless, we can use the techniques here in many applications. With some extension and modification, the proposed schemes can also help recover missing data chunks when the content is

transmitted through imperfect transmission channels. Future work includes how to design the error-recovery system with maximum recovery ability and minimum additional bandwidth and computational power requirements.

References

1. I.J. Cox, "A Secure, Robust Watermark for Multimedia," *Proc. Information Hiding 96*, Lecture Notes in Computer Science, vol. 1174, Springer-Verlag, Berlin, pp. 185-206.
2. I.J. Cox and M.L. Miller, "A Review of Watermarking and the Importance of Perceptual Modeling Human Vision and Electrtonic Imaging II," *Proc. SPIE 3016*, SPIE Press, Bellingham, Wa., 1997, pp. 92-99.
3. C.W. Wu et al., "Fragile Imperceptible Digital Watermark with Privacy Control," *Electronic Imaging 99 Security and Watermarking of Multimedia Content, SPIE 3657*, SPIE Press, Bellingham, Wa., 1999, pp. 79-84.
4. J. Brassil et al., "Electronic Marking and Identification Techniques to Discourage Document Copying," *Proc. IEEE Infocom 94*, IEEE Press, Piscataway, N.J., pp. 1278-1287.
5. J. Brassil et al., "Hiding Information in Documents Images," *Proc. Conf. Information Sciences and Systems (CISS-95)*, 1995, pp.482-489.
6. Y. Liu et al., "Marking and Detection of Text Documents using Transform-domain Techniques," *Proc. Electronic Imaging 99*, *Security and Watermarking of Multimedia Content, SPIE 3657*, SPIE Press, Bellingham, Wa., 1999, pp. 317-328.
7. W. Bender et al., "Techniques for Data Hiding," *IBM Systems J.*, vol. 35, no. 3-4, 1996, pp. 313-336.
8. L. Boney et al., "Digital Watermarks for Audio Signals," *IEEE Int'l Conf. Multimedia Computing and Systems*, IEEE CS Press, Los Alamitos, Calif., 1996, pp. 473-480.
9. J. Dittmann, M. Stabenau, and R. Steinmetz, "Robust MPEG Video WatermarkingTechnologies," *Proc. Multimedia 98*, ACM Presss, New York, 1998, pp. 71-80.
10. C.Y. Lin and S.F, "Chang Issues and Solutions for Authenticating MPEG Video," *Electronic Imaging '99. Security and Watermarking of Multimedia Content, SPIE 3657*, SPIE Press, Bellingham, Wa., 1999, pp.54-65.
11. H. Yu, "Content-Based Graph Authentication," *Proc. ACM Multimedia 98 Multimedia Security Workshop*, GMD, 1999, pp. 101-108.
12. H. Yu, "Document Authentication," submitted for publication.
13. H. Yu, "Intelligent System for Symbolic Representation of Graph," submitted for publication.

Heather Yu is a senior research scientist working on secure e-commerce projects at the Panasonic Information and Networking Technologies Laboratory. She is also the committee secretary of IEEE Communications Society's Multimedia Communiccations Technical Committee and an associate editor for the *IEEE Transactions on Multimedia*. Her research interests include multimedia signal processing, media security, secure digital content access and distribution, and multimedia content retrieval. She has a BS in electrical engineering from Peking University, Beijing, and an MA and PhD in electrical engineering from Princeton University.



Wayne Wolf is a professor of electrical engineering at Princeton University. His research include multimedia systems, embedded computing, and VLSI. He has a BS, MS, and PhD in electrical engineering from Stanford University. He is an IEEE fellow and an ACM and SPIE member.

Xiangyang Kong is a software developer at the Panasonic Information and Networking Technologies Laboratory. His research interests include multimedia computing and security. In the past two years, he implemented several software demonstration packages on media security. He has an MS in computer science from the New Jersey Institute of Technology and a PhD in molecular genetics from Texas A&M University.

Readers may contact Yu at Panasonic Information and Networking Technology Lab, Panasonic, 2 Research Way, Princeton, NJ 08540, email heathery@research.panasonic.com.