Quotation Authentication: A New Approach and Efficient Solutions by Cascaded Hashing Techniques

Tsung-Yuan Liu, *Student Member, IEEE*, and Wen-Hsiang Tsai, *Senior Member, IEEE*

***Abstract*—A new approach to efficient authentication of quotations from trusted sources but embedded in messages created by unknown authors is proposed. Two different techniques for efficient generation of authenticable quotations using cascaded hashing are proposed for conventional quotations, and an efficient two-dimensional signature generation technique is proposed for two-dimensional quotations from spreadsheets. A technique for associating the signature with the quotation in the Microsoft Word document is also proposed for generation of integrated authenticable quotations that survive copy-and-paste operations.**

***Index Terms*—Authenticable quotation, information security, Microsoft Excel, Microsoft Word, quotation authentication.**

I. Introduction

THE advancement of digital technology and the Internet has greatly eased the publication of information to the mass, whether in the form of web pages, e-mails, PDF files, or office documents. The abundance of content made available is mostly beneficial to the general public, but significant amounts of annoying or even harmful information are also being distributed on the Internet.

There are many ways to help identify useful information. Search engines, such as Google and Yahoo, analyze the interlinkage of documents on the web and deduce the relative degrees of importance of the documents by using their page ranking technology [1], with the result that documents ranked high in their search results tend to be more trustworthy. Also, people tend to trust information published by *credible sources*, for example, announcements by federal or local governments, publications from the IEEE, ACM, or other famous publishers with stringent peer review processes, news reported by CNN, ABC, or other well-known news agencies, and so on.

There are, however, numerous information publishers and distributors which are less known to the public and yet provide useful information. In particular, they sometimes organize valuable content published by credible sources by quoting, aggregating, or disseminating them to make the edited data more relevant or accessible to readers. For example, it is common for an article to quote significant findings from technical journals or government reports that would otherwise never be read by the general public.

Unfortunately, there is no easy way for a reader to verify that the quotations contained in these articles are *really* from the claimed sources. This vulnerability is frequently used in *Internet hoaxes*, such as virus warnings and sympathy letters. One mobile phone virus hoax, for example, claims that if a user answers a call without caller identity, the user's phone can be infected and no longer usable. The hoax lures the reader by claiming that the information has been confirmed by credible mobile phone companies (like Nokia and Motorola), and that the news was reported on CNN. Due to the credibility of the claimed sources, most people forwarded the message without performing any further verification. The multiplicative effect of such blind message passing puts unnecessary burdens on the network and servers, and costs the reader valuable time to process them. Another example is nutrition and health recommendations that point out unhealthy food, products, or life styles. The contents are mixed with myths, true pieces of information, forged allegations on competitor companies, and so on. Such content comes from a variety of sources, some of which may not be accessible online or require special subscriptions, making it difficult for a message reader to verify the truthfulness of the information.

In this study, we investigate this problem of *quotation authentication*, where a new approach is proposed to allow efficient authentication of quotations from trusted sources but embedded in documents created by unknown authors. The three parties involved in quotation authentication are, respectively, the *source author* A^S of a source article S, the *message author* A^M of a message M that contains quotations from S, and the *message reader* R^M of M. We assume that R^M recognizes A^S as a credible source but does not know A^M. The goal of quotation authentication is to allow R^M to efficiently and undisputedly *verify* the fidelity and source of the quotations contained in M.

The importance to verify quotations within a message was recognized by Fernstrom [2]. However, assumed in the proposed solutions is either that the message reader has access to the original source article or that there is a trusted party to which message authors can send arbitrary quotations and source articles in order to get endorsed quotations that are verifiable by the message reader. The approach proposed in this paper to solve the quotation authentication problem is instead more practical, where each of the three parties only needs to perform certain simple and deterministic processes, as described by the following scenario:

1) A^S processes the source S to create a certain *source signature* G^S, and then publishes S along with G^S to the general public;

Manuscript received March 01, 2010; revised July 15, 2010; accepted August 24, 2010. Date of publication September 02, 2010; date of current version November 17, 2010. This work was supported in part by the NSC project 97-2631-H-009-001. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Darko Kirovski.

The authors are with the Department of Computer Science, National Chiao Tung University, Hsinchu 300, Taiwan (e-mail: gis91811@cis.nctu.edu.tw; whtsai@cis.nctu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2010.2072501

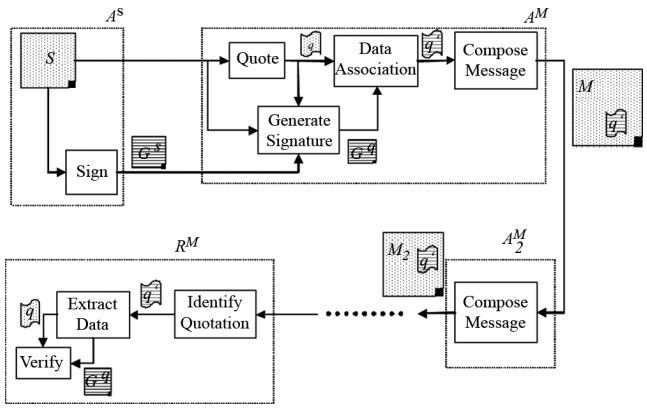


Fig. 1. Illustration of processes performed by and information passed between a source author, one or more message authors, and a message reader.

2) A^M cites a text segment q of interest within S, uses G^S to create an appropriate *quotation signature* G^q for q, creates an integrated *authenticable quotation* q' by associating G^q with q, and puts q' into M;
3) finally R^M, when reading M, verifies that the quotation q, which is contained in q', is indeed authored by A^S.

A benefit of associating the signature with the corresponding quotation in Stage 2 above to create an integrated authenticable quotation q' is that the quotation can be distributed multiple times using copy-and-paste operations by several message authors without hindering the ability for the final message reader to identify such authenticable quotations and to verify that the quotation is indeed authored by A^S. The idea of the proposed approach is illustrated in Fig. 1. And the technique of associating the signature data with the quotation is referred to hereafter as a *data association* technique, the details of which will be described in Section V.

It is assumed in the above proposed method that an appropriate quotation signature G^q can be generated using the source signature G^S, thus alleviating the need for a third-party to endorse quotations. Also assumed is that G^q alone is sufficient for R^M to verify that the quotation q is indeed from A^S, so that R^M does *not* have to access the original source S. Although the digital signatures G^S and G^q, attached to S and q, respectively, are the overhead required to achieve the goal of quotation authentication, we propose in this study techniques for generating appropriate source and quotation signatures to minimize the sizes of the overhead.

In the remainder of the paper, related work on message authentication is reviewed in Section II. In Section III, two basic techniques to generate source and quotation signatures are first described followed by the proposed more efficient techniques that minimize the sizes of the signatures required. Quotations that are not consecutive, in particular two-dimensional quotations from spreadsheet documents, are discussed in Section IV, and an efficient method for generating source and quotation signatures for the two-dimensional case is described. Presented in Section V are the proposed data association techniques for two popular document formats, Microsoft Word and Microsoft Excel, to demonstrate the feasibility of the proposed method. Finally, we conclude in Section VI and point out some possible future work.

II. Related Work

Methods to authenticate a message *as a whole* are relatively well studied. Typically, two communicating parties are assumed to share a secret key that is used to generate an appropriate message authentication code (MAC) for verifying the fidelity of content transmitted over an insecure channel. Bellare *et al.* [3], [4] proposed the keyed-hash message authentication code (HMAC) that can generate provably secure message authentication codes based on standard hash functions such as MD5 and SHA1. Cipher-based MAC (CMAC) techniques such as XMAC and OMAC use standard block ciphers such as DES and AES to construct message authentication codes [5].

In the *asymmetric* case, a sender creates, using its *private key*, an appropriate *digital signature* for the transmitted content, such that the receiver can verify the fidelity of the content using the *public key* of the sender. The most straightforward way to generate a digital signature is to generate a hash value for the content and encrypt the value using asymmetric encryption such as the RSA algorithm, as described in PKCS#1 [6]. A related research area is digital certificate works, which provide means for associating the human recognizable identity of an entity with that entity's public key. The HTTPS scheme is an example of such works where websites register and use X.509 certificates that are issued by trusted certificate authorities [7], such that users can authenticate the websites they connect to.

There is also a branch of research in stream authentication [8], where receivers need to verify whether the received stream of packets are from the intended source. The problem is made complex when dealing with authenticating multicast traffic in the face of packet losses. Efficient schemes for multicast authentication have been proposed by using the temporal relationships of network packets, that is, later packets can be used to verify the authenticity of prior packets [9]. These techniques, however, cannot be applied to the problem of quotation authentication studied here since there is no notion of time in messages and quotations.

III. Signature Generation Techniques for Quotation Authentication

As mentioned previously, the proposed approach to quotation authentication requires the three parties involved to respectively generate an appropriate source signature, generate an appropriate quotation signature, and verify the quotation signature. We describe in this section two basic techniques to generate source and quotation signatures and point out their shortcomings. The proposed more efficient techniques are then described subsequently.

A. Basic Quotation Authentication Techniques

A basic quotation authentication technique is for the source author to generate a digital signature for *every* possible quotation in a source article and to publish the signatures along with the source article on a certain website. A message author then only needs to quote the desired text and bundle it with the associated published digital signature to generate a valid authenticable quotation. This technique is, however, inefficient since a quotation may start and end at any position of a document, and so the number of possible quotations required for a document

of length L is, as can be figured out, of the complexity order $O(L^2)$. This means in turn that the number of digital signatures is also of the order $O(L^2)$. This scheme is referred to hereafter as the *enumerate-all-quotations* technique.

Another basic technique is to include the *entire* source article S, together with an appropriate digital signature, into the quotation signature G^q of a quotation q. A message reader can then easily extract S from G^q and verify that the quotation really comes from S. However, including the whole article S will bloat the size of G^q to be of the order $O(L)$ because S here is an overhead which is not needed in the enumerate-all-quotations technique. Also, the act of attaching the entire article S might violate the copyright law. This scheme is referred to hereafter as the *quote-the-whole* technique.

B. Overhead of Quotation Authentication Techniques

As mentioned previously, the overhead of quotation authentication is the extra source and quotation signatures needed to allow generation and verification of authenticable quotations. Assume that texts are quoted from S by P message authors to generate P messages, which are consumed by Q message readers. The size of the total overhead of all the source signatures G^S and the quotation signatures G^q for such a quotation authentication process, called *total overhead size*, is $P|G^S| + P|G^q| + Q|G^q|$, where $|\cdot|$ means the size of "$\cdot$", since each of the P message authors needs to receive S as well as G^S from the source author and include q as well as G^q in his/her message, and each message reader needs to receive q and G^q from a message author. Note that S and q are not included in the overhead.

Consequently, the total overhead size of the enumerate-all-quotations technique is of $O(PL^2 + P + Q)$, or equivalently, $O(PL^2 + Q)$ since G^S contains $O(L^2)$ digital signatures and G^q contains a single digital signature. On the other hand, the total overhead size of the quote-the-whole technique is of $O(P + PL + QL)$, or equivalently, $O(PL + QL)$, since G^S is a single digital signature, but the size of G^q, which includes S as an overhead as mentioned previously, is of the order $O(L)$.

If Q is significantly larger than P, that is, if there are many more message readers than message authors, then the enumerate-all-quotations technique is more efficient than the quote-the-whole one in view of the total overhead size. However, if the magnitudes of P and Q are comparable, then the quote-the-whole technique is more efficient. Since the two techniques have their respective merits for different application cases, we propose in this study two better techniques which improve them respectively with the same goal of minimizing the total overhead size, as described in the following.

C. Multiuse Signatures Technique (MUST)

The basic idea of the enumerate-all-quotations technique is to include in the source signature G^S the digital signatures for all possible quotations so that the size of the quotation signature G^q is small for any quotation. This is, however, overly inefficient for large S, and we describe in the following a *multiuse signatures technique* (MUST) where the size of G^q remains to be of the order $O(1)$ for all quotations while the size of G^S required is reduced to be of the order $O(L)$ instead of $O(L^2)$.

Assume without loss of generality that S consists of L *concatenated sentences* denoted as $s_1||s_2||s_3||\dots||s_L$ where "$||$" specifies the concatenation operator, and the length of each sentence is *bounded*, meaning that each sentence s_i is of the order $O(1)$ in size. Also assumed is that the quotations include *complete* sentences, that is, q includes a set of consecutive sentences in S such that $q = s_a||\dots||s_b$, where $1 \le a \le b \le L$. If this is not the case, the portion of text to complete a sentence can be included in the quotation signature (this is described in more detail in Section V).

The idea of the proposed MUST is for the source author to generate L *multiuse signatures* g_j, where $1 \le j \le L$, such that the signature g_j can be used as part of the quotation signature for any of the quotations $s_a||s_{a+1}||\dots||s_j$, where $1 \le a \le j$. In addition to the multiuse signatures, we also generate L *cascaded hash values* j_j, where $1 \le j \le L$, and include the hash value h_j into the quotation signature for a quotation $s_j||s_{j+1}||\dots||s_b$, where $j \le b \le L$. The generation of the multiuse signatures and cascaded hash values is described in detail in the following algorithm.

Algorithm 1: Generation of a MUST Source Signature

Input: a source article S consisting of L sentences s_j, where $1 \le j \le L$.

Output: a source signature G^S, which contains L cascaded hash values h_j and L multiuse signatures g_j, where $1 \le j \le L$.

Steps:

1) Set the first cascaded hash value to be the hash value of the entire source, that is, set $h_1 = H(S)$, where $H(\cdot)$ is some hash function.
2) For each j from 2 to L, compute the cascaded hash value h_j as $h_j = H(h_{j-1}||H(s_{j-1}))$.
3) For $1 \le j \le L$, calculate the multiuse signature g_j as $g_j = \text{Sign}(h_j||H(s_j))$, where Sign $(\cdot)$ is a signing function of some digital signature algorithm.

As a simple example, the cascaded hash values generated by the above algorithm for a source article with three sentences is as follows: $h_1 = H(S)$, $h_2 = H(h_1||H(s_1))$, and $h_3 = H(h_2||H(s_2))$. And the quotation signature for the quotation $q = s_3$ contains h_3 and g_3, where $g_3 = \text{Sign}(h_3||H(s_3))$. Note that in detail h_3 equals $H(h_2||H(s_2)) = H(H(h_1||H(s_1))||H(s_2)) = H(H(H(S)||H(s_1))||H(s_2))$, which is a value derived by a cascade of concatenation and hashing operations, hence the name cascaded hash value.

For a quotation $q = s_a||s_{a+1}||\dots||s_b$, the quotation signature G^q includes h_a and g_b. When a message reader receives a quotation q and its quotation signature G^q, the following algorithm is proposed to verify q.

Algorithm 2: Quotation Verification Using MUST

Input: a quotation $q = s_a||s_{a+1}||\dots||s_b$, where $a \le b$, a cascaded hash value h_a, and a multiuse signature g_b.

Output: the result of quotation verification.

Steps:

1) Perform the following steps to compute the cascaded hash values $h_{a+1}, h_{a+2}, \ldots, h_b$ using q and h_a:
   a) initialize the value i to be a;
   b) compute the cascaded hash value h_{i+1} to be $H(h_i||H(s_i))$;
   c) increment the value of i by 1, and go to Step 1b if i is smaller than b.
2) Verify the quotation by Verify $(h_b||H(s_b), g_b)$ and output the result, where Verify is the reciprocal digital signature verification function of the Sign function used by the source author.

The above verification works for any quotation $q = s_a||s_{a+1}||\ldots||s_b$, where $1 \le a \le b \le L$, because the cascaded hash value h_b calculated by Algorithm 2 using h_a and q is the same as that calculated in Algorithm 1. This is easily seen since the computations performed by Step 1b of Algorithm 2 are the same as those by Step 2 of Algorithm 1.

Since the cascaded hash value h_b is constructed by using cascaded operations of hashing and concatenation, an attacker needs to find exact collisions of hash values in order to craft a forged quotation as well as a matching quotation signature that can ensure the same h_b to be constructed using Algorithm 2. The proposed technique is thus attack-resilient if the functions H and Sign are chosen properly, such as using, respectively, the hash function SHA1 and the signing function RSA with sufficiently long keys [5], [10].

A MUST source signature G^S for a source article S with L sentences always contain exactly L cascaded hash values and L multiuse signatures, and so the size of G^S is of the order $O(L)$. A MUST quotation signature G^q always contains one cascaded hash value and one multiuse signature, and thus the size of G^q is of the order $O(1)$. The total overhead size of the MUST with P message authors and Q message readers is thus $O(PL+P+Q)$, or equivalently, $O(PL+Q)$, contrasted with the originally larger total overhead size of $O(PL^2+Q)$ for the basic enumerate-all-quotations technique.

D. Tree Root Uni-Signature Technique (TRUST)

The previously described quote-the-whole technique requires the complete source article S to be included in the G^q of each q. This is overly inefficient for a large S, and the proposed improving technique, called *tree root uni-signature technique* (TRUST), is described in this section. The TRUST uses a *tree-like* construction of hash values such that the size of G^q can be reduced to be of the order $O(\log_2 L)$. *Only the root* of the tree of the hash values needs to be signed by the source author and kept as an overhead, thus maintaining the size of G^S to be of $O(1)$.

We assume as before that S consists of L sentences and that quotations include complete sentences. In the proposed TRUST, the source author generates a binary tree of hash values h_j^i from S by the following algorithm, where i is the depth of the tree node and j is the index of the nodes at depth i. The hash value of the tree root h_1^r, where $r = \lceil \log_2 L \rceil + 1$, is then signed with some digital signature algorithm to get the *tree root uni-signature* $g_1^r = \mathrm{Sign}(h_1^r)$. It is noted that this part of signature generation is similar to that of a traditional Merkle tree.

TABLE I
TRUST TREE OF HASH VALUES FOR FIVE SENTENCES

i	Hash values					$\lvert h^i \rvert$
1	$h^1_1 = \mathrm{H}(s_1)$	$h^1_2 = \mathrm{H}(s_2)$	$h^1_3 = \mathrm{H}(s_3)$	$h^1_4 = \mathrm{H}(s_4)$	$h^1_5 = \mathrm{H}(s_5)$	5
2	$h^2_1 = \mathrm{H}(h^1_1 \lVert h^1_2)$		$h^2_2 = \mathrm{H}(h^1_3 \lVert h^1_4)$		$h^2_3 = h^1_5$	3
3	$h^3_1 = \mathrm{H}(h^2_1 \lVert h^2_2)$				$h^3_2 = h^2_3$	2
4	root: $h^4_1 = \mathrm{H}(h^3_1 \lVert h^3_2)$					1

Algorithm 3: Generation of a TRUST Tree of Hash Values

Input: a source article S consisting of L sentences $s_1, s_2, \ldots, s_L$.

Output: a tree of hash values h_j^i, where $1 \le i \le r$, and j is the index of the nodes at depth i.

Steps:

1) Calculate a hash value for each sentence s_j to get the lowest-level hash values, that is, set $h_j^1 = H(s_j)$ for $1 \le j \le L$, where $H(\cdot)$ is some hash function.
2) Concatenate the hash values in pairs and compute the second-level hash values as $h_j^2 = H(h_{2j-1}^1||h_{2j}^1)$ for $1 \le j \le \lfloor L/2 \rfloor$, where $\lfloor \cdot \rfloor$ means the floor operation. If L is odd, take the final hash value for $j = (L+1)/2$ to be $h_j^2 = h_{2j-1}^1$.
3) Repeat the above step to calculate the third-level hash values $h_j^3 = H(h_{2j-1}^2||h_{2j}^2)$ and so on, each time with the number of calculated hash values being equal to a half of the number of those computed in the previous step (plus one if odd), until finally the hash value h_1^r is generated for the tree root.

As an illustration, the TRUST tree of hash values generated by the above algorithm for $L = 5$ is shown in Table I below. The number of hash values at each level i is denoted as $|h^i|$.

For any quotation q from S, a TRUST quotation signature contains the signature g_1^r generated by the source author and a quotation-dependent set H^q, called the *complementary hash set*, of some of the hash values generated by Algorithm 3 above. In more detail, for a quotation $q = s_a||s_{a+1}||\ldots||s_b$ where $1 \le a \le b \le L$, the complementary hash set is selected using the algorithm below.

Algorithm 4: Generation of a TRUST Complementary Hash Set

Input: a source article $q = s_1||s_2||\ldots||s_L$ and a quotation $q = s_a||s_{a+1}||\ldots||s_b$.

Output: a complementary hash set H^q for q.

Steps:

1) Calculate the TRUST tree of hash values h_j^i for S using Algorithm 3.

2) Set the hash set H^q to be the empty set initially, and set the value of i to be 1.
3) If a is even, then add the value h^i_{a-1} to H^q.
4) If b is odd and is smaller than $|h^i|$, then add the value h^i_{b+1} to H^q.
5) Set a to be $\lceil a/2 \rceil$ and b to be $\lceil b/2 \rceil$, where $\lceil \cdot \rceil$ means the ceiling operation.
6) Increment the value of i by 1, and go to Step 3 if i is smaller than $r = \lceil \log_2 L \rceil + 1$.

As an example, if S has five sentences and we wish to quote the second and third sentences (i.e., $q = s_2 || s_3$), then according to Algorithm 4, the value of i is set to 1 initially, and is increased to 2, then to 3, and finally to 4; the value of a decreases from 2 to 1, and remains at 1; and the value of b decreases from 3 to 2, and then to 1. And so the hash values h^1_1, h^1_4, and h^3_2 are added to H^q. It can be seen that the hash value of the tree root h^r_1 can be reconstructed as follows using only H^q and q:

$$\begin{aligned} h^r_1 &= H(h^3_1 || h^3_2) = H(H(h^2_1 || h^2_2) || h^3_2)) \\ &= H(H(H(h^1_1 || h^1_2)) || H(h^1_3 || h^1_4)) || h^3_2) \\ &= H(H(H(h^1_1 || H(s_2))) || H(H(s_3) || h^1_4)) || h^3_2). \end{aligned}$$

This property may be generalized for any quotation of a source article, and used for quotation verification, as described in the following.

For a quotation $q = s_a || s_{a+1} || \ldots || s_b$, the quotation signature G^q includes the values a, b, and L, the signature g^r_1, and the complementary hash set H^q. When a message reader receives a quotation q and its quotation signature G^q, the following algorithm is proposed to reconstruct the tree-root hash value using q and H^q and to verify the quotation q using g^r_1.

Algorithm 5: Quotation Verification Using the TRUST

Input: a quotation $q = s_a || s_{a+1} || \ldots || s_b$, where $a \le b$; a tree root uni-signature g^r_1; a complementary hash set H^q; and the values a, b, and L.

Output: the result of quotation verification.

Steps:

1) Set the value of i to be 1, and calculate the lowest-level hash values for the sentences s_a through s_b in the quotation, that is, set $h^1_j = H(s_j)$ for $1 \le j \le b$.
2) If a is even, then retrieve the next value from H^q, which is h^i_{a-1}.
3) If b is odd and is smaller than $|h^i|$, then retrieve the next value from H^q, which is h^i_{b+1}.
4) Set a to be $\lceil a/2 \rceil$ and b to be $\lceil b/2 \rceil$.
5) Compute the next-level hash values $h^{i+1}_j = H(h^i_{2j-1} || h^i_{2j})$ for $a \le j \le b$.
6) Increment the value of i by 1, and go to Step 3 if i is smaller than $r = \lceil \log_2 L \rceil + 1$.
7) Verify the quotation by Verify (h^i_1, g^r_1), where h^i_1 is the tree root hash value and output the result.

Why H^q is called the complementary hash set should now be clear from the above algorithm—it fills in where needed such that the next-level hash values can be calculated from the sequence of hash values $h^i_a, h^i_{a+1}, \ldots, h^i_b$. The process continues until the final tree root hash value h^r_1 is computed. Since h^r_1 is constructed by repeated applications of hashing followed by concatenation, it is difficult for an attacker to craft a forged quotation and a corresponding hash set that can ensure the same value of h^r_1 to be constructed in the same way as that in the case of a legitimate quotation.

A TRUST source signature always contains just a single digital signature g^r_1, and is thus of the order $O(1)$. On the other hand, a TRUST complementary hash set H^q has no more than $2(r-1)$ hash values since hash values are added to H^q only during Steps 2 and 3 of Algorithm 5 and the steps are repeated for exactly $(r-1)$ times. Hence, the size of H^q, and also that of G^q, have an order of $O(\log_2 L)$ because $r = \lceil \log_2 L \rceil + 1$. The total overhead size of the TRUST is thus $O(P + P \log_2 L + Q \log_2 L)$, or equivalently, $O(P \log_2 L + Q \log_2 L)$, contrasted with the total overhead size of $O(PL + QL)$ for the basic quote-the-whole technique.

	A	B	C	D	E	F	G	H
1	**Revenues**	2002	2003	2004	2005	2006	2007	2008
2	**Revenues**	439,508	1,465,934	3,189,223	6,138,560	10,604,917	16,593,986	21,795,550
3	**Google web sites**	306,978	792,063	1,589,032	3,377,061	6,332,797	10,624,705	14,413,826
4	**Google Network web sites**	103,937	628,600	1,554,256	2,687,942	4,159,831	5,787,938	6,714,688
5	**Total Advertising Revenues**	410,915	1,420,663	3,143,288	6,065,003	10,492,628	16,412,643	21,128,514
6	**Licensing & other revenues**	28,593	45,271	45,935	73,558	112,289	181,343	667,036
7	**As % of Revenues**							
8	*Google web sites*	*70%*	*54%*	*50%*	*55%*	*60%*	*64%*	*66%*
9	*Google Network web sites*	*24%*	*43%*	*49%*	*44%*	*39%*	*35%*	*31%*
10	*Licensing & other revenue*	*6%*	*3%*	*1%*	*1%*	*1%*	*1%*	*3%*

Fig. 2. Two-dimensional quotation in a spreadsheet document (source: Google Investor Relations).

IV. Quotation Authentication for Spreadsheet Documents

All discussions up until now assume that the documents and quotations are texts that flow *consecutively*. However, the contents in some documents are not sequential texts. In particular, a spreadsheet such as a Microsoft Excel document contains sheets of two-dimensional data or *cells*. We describe possible quotations from such a kind of document in the following, and propose a technique to efficiently generate source and quotation signatures.

A. Two-Dimensional Quotations

A typical quotation from a spreadsheet document is not sequential cells but a two-dimensional cutout of cells. For example, a company's income statement may list the details of the revenue and expense figures in rows, with the values for different years or quarters across columns, as shown in Fig. 2. If a business analyst (analogous to the message author in our problem of quotation authentication) quotes the figures of the revenues for a few selected years, the selection would be a two-dimensional subset of the spreadsheet, as illustrated in the figure.

The basic techniques described in Section III-A can be applied to the two-dimensional case here, but are inefficient. For a spreadsheet containing X columns by Y rows, it can be shown that the total overhead size for the enumerate-all-quotations and the quote-the-whole techniques are $O(P(XY)^2 + Q)$ and $O(PXY + QXY)$, respectively.

The proposed improving techniques of the MUST and the TRUST can be used in the two-dimensional case by generating a quotation signature for each row in the quotation. However, this means that for a quotation that contains, say, B rows, B separate quotation signatures will be needed. That is, the total overhead size is proportional to the size of the two-dimensional quotation. We describe below a better technique that yields a total overhead size of $O(P\log_2 XY + Q\log_2 XY)$ irrespective of the quotation size.

B. Two-Dimensional Tree Root Uni-Signature Technique

The proposed improving technique, called the *two-dimensional tree root uni-signature technique* (2-D TRUST), uses a tree-like construction of hash values similar to that of the previously described 1-D TRUST such that the size of the quotation signatures can be confined to be of the order $O(\log_2 XY)$. Only the root of the two-dimensional tree of the hash values needs to be signed by the source author, thus maintaining the size of the source signature to be of the order $O(1)$.

In a source spreadsheet document S consisting of X columns and Y rows of cells, denote the cell at column x and row y by $s_{x,y}$, where $1 \le x \le X$ and $1 \le y \le Y$. The proposed technique requires the source author to generate a two-dimensional tree of hash values for the cells in S, as described by the following algorithm. The hash value $h^r_{1,1}$ of the tree root is then signed with some digital signature algorithm to get a *two-dimensional tree root uni-signature* $g^r_{1,1} = \mathrm{Sign}(h^r_{1,1})$.

Algorithm 6: Generation of a 2-D TRUST Tree of Hash Values

Input: a source spreadsheet S consisting of $X \times Y$ cells $s_{x,y}$, where $1 \le x \le X$ and $1 \le y \le Y$.

Output: a two-dimensional tree of hash values $h^i_{x,y}$, where i is the depth of the tree node; x and y are the indexes of the nodes at depth i; and $h^r_{1,1}$ is the hash value of the tree root.

Steps:

1) Calculate a hash value for each cell $s_{x,y}$ to get the lowest-level hash values, that is, set $h^1_{x,y} = H(s_{x,y})$ for $1 \le x \le X$ and $1 \le y \le Y$.
2) Initialize the value of i to be 1.
3) For all values of x and y, where $1 \le x \le \lfloor X/2 \rfloor$ and $1 \le yx \le \lfloor Y/2 \rfloor$, concatenate the hash values in fours to compute the next-level hash values as $h^{i+1}_{x,y} = H(h^i_{2x-1,2y-1} \| h^i_{2x-1,2y} \| h^i_{2x,2y-1} \| h^i_{2x,2y})$.
4) Perform one of the following operations, depending on the values of X and Y:
   a) if both X and Y are even, then set X to be $X/2$ and Y to be $Y/2$;
   b) if X is odd and Y is even, then set X to be $(X+1)/2$ and Y to be $Y/2$; and set $h^{i+1}_{X,Y} = \mathrm{H}(h^i_{X-1,2Y-1} \| h^i_{2X-1,2Y})$;
   c) if X is even and Y is odd, then set X to be $X/2$ and Y to be $(Y+1)/2$; and set $h^{i+1}_{X,Y} = \mathrm{H}(h^i_{X-1,2Y-1} \| h^i_{2X,2Y-1})$;
   d) if both X and Y are odd, then set X to be $(X+1)/2$ and Y to be $(Y+1)/2$; and set $h^{i+1}_{X,Y} = h^i_{2X-1,2Y-1}$.
5) Increment the value of i by 1.
6) Denote the numbers of columns and rows of hash values for level i as $X^i = X$ and $Y^i = Y$, respectively.
7) Go to Step 3 if the tree root hash value has not been calculated, or equivalently, if X or Y is larger than 1.
8) Set the total number of levels r to be i.

It is not difficult to figure out from the steps of the above algorithm that r computed in Step 8 is $r = \max(\lceil \log_2 X \rceil, \lceil \log_2 Y \rceil) + 1$.

When a message author A^M quotes a two-dimensional rectangle of cells in a source spreadsheet, the quotation signature G^q is generated by including the signature $g^r_{1,1}$ and a set of complementary hash values H^q containing some of the hash values generated in Algorithm 6. For a quotation q that contains a rectangle of cells with a top-left cell $s_{a,b}$ and a bottom-right one $s_{c,d}$, where $1 \le a \le c \le X$ and $1 \le b \le d \le Y$, the complementary hash set is selected using the algorithm below. The purpose of H^q is the same as that in the 1-D TRUST, that is, the tree root hash value $h^r_{1,1}$ can be reconstructed from q and H^q.

Algorithm 7: Generation of a 2-D TRUST Complementary Hash Set

Input: a source spreadsheet S consisting of $X \times Y$ cells $s_{x,y}$, where $1 \le x \le X$ and $1 \le y \le Y$, and a two-dimensional rectangle of quoted cells q with a top-left cell $s_{a,b}$ and a bottom-right one $s_{c,d}$, where $1 \le a \le c \le X$ and $1 \le c \le d \le Y$.

Output: a complementary hash set H^q.

Steps:

1) Calculate from S the tree of hash values $h^i_{x,y}$ using Algorithm 6, with $X^i \times Y^i$ hash values generated for level i.
2) Set H^q to be the empty set initially, and set the value of i to be 1.
3) Add to H^q the following hash values so that the next-level hash values can be reconstructed from q:
   a) if a is even and b is odd, then add the values $h^i_{a-1,b}$ and $h^i_{a-1,b+1}$ to H^q;
   b) if a is odd and b is even, then add the values $h^i_{a,b-1}$ and $h^i_{a+1,b-1}$ to H^q;
   c) if both a and b are even, then add the values $h^i_{a-1,b-1}$, $h^i_{a,b-1}$ and $h^i_{a-1,b}$ to H^q;
   d) if c is even, d is odd, and $d < Y^i$, then add the values $h^i_{c-1,d+1}$ and $h^i_{c,d+1}$ to H^q;
   e) if c is odd, d is even, and $c < X^i$, then add the values $h^i_{c+1,d-1}$ and $h^i_{c+1,d}$ to H^q;
   f) if both c and d are odd, then:
      i) add the value $h^i_{c+1,d}$ to H^q if $c < X^i$;
      ii) add the value $h^i_{c,d+1}$ to H^q if $d < Y^i$;
      iii) add the value $h^i_{c+1,d+1}$ to H^q if $c < X^i$ and $d < Y^i$.

TABLE II
HASH VALUES SELECTED INTO THE 2-D TRUST COMPLEMENTARY HASH SET WHEN QUOTING A CELL $s_{3,2}$ FROM A5 × 5 SPREADSHEET

	1	2	3	4	5
1	$h^2_{1,1}$		$h^2_{1,2}$		$h^3_{1,2}$
2					
3	$h^1_{3,1}$	$s_{3,2}$	$h^2_{2,2}$		
4	$h^1_{4,1}$	$h^1_{4,2}$			
5	$h^3_{2,1}$				$h^3_{2,2}$

4) Set a to be $\lceil a/2 \rceil$, b to be $\lceil b/2 \rceil$, c to be $\lceil c/2 \rceil$, and d to be $\lceil d/2 \rceil$.
5) Increment the value of i by 1, and go to Step 3 if i is smaller than $r = \max(\lceil \log_2 X \rceil, \lceil \log_2 Y \rceil) + 1$.

As a simple example, when quoting a single cell $s_{3,2}$ from a 5×5 spreadsheet, we have $a = c = 3$ and $b = d = 2$, and we add the values $h^1_{3,1}$, $h^1_{4,1}$, and $h^1_{4,2}$ to H^q when i is 1 (in Steps 3b and 3e). The values of a and c are then set to be 2 while the values of b and d are set to be 1 in Step 4. In the next iteration, the values $h^2_{1,1}$, $h^2_{1,2}$, and $h^2_{2,2}$ are added to H^q when i is 2 (in Steps 3a and 3d), and the values of a, b, c, and d are all set to be 1. In the last iteration, the hash values $h^3_{2,1}$, $h^3_{1,2}$, and $h^3_{2,2}$ are added to H^q when i is 3 (in Step 3f). The hash values added to H^q for this example is illustrated in Table II below.

For a two-dimensional quotation q with a top-left cell $s_{a,b}$ and a bottom-right one $s_{c,d}$, the quotation signature G^q includes the signature $g^r_{1,1}$, the complementary hash set H^q, and the values X and Y (from which the value r and the values X^i and Y^i, where $1 \le i \le r$, can be calculated). When a message reader receives a quotation q and its quotation signature G^q, the following algorithm is proposed to reconstruct the tree-root hash value $h^r_{1,1}$ using q and H^q and to verify the quotation q using $g^r_{1,1}$.

Algorithm 8: Verification of a 2-D TRUST Quotation Signature

Input: a two-dimensional quotation with a top-left cell of $s_{a,b}$ and a bottom-right one of $s_{c,d}$, where $a \le c$ and $b \le d$; a signature $g^r_{1,1}$; a complementary hash set H^q; and the values X and Y.

Output: the result of quotation verification.

Steps:
1) Set the value of i to be 1, and calculate the lowest-level hash values for the cells in the quotation, that is, set $h^1_{x,y} = \mathrm{H}(s_{x,y})$ for $a \le x \le c$ and $b \le y \le d$.
2) Retrieve the following values from H^q to calculate the next-level hash values:
   a) if a is even and b is odd, then retrieve the values $h^i_{a-1,b}$ and $h^i_{a-1,b+1}$ from H^q;
   b) if a is odd and b is even, then retrieve the values $h^i_{a,b-1}$ and $h^i_{a+1,b-1}$ from H^q;
   c) if both a and b are even, then retrieve the values $h^i_{a-1,b-1}$, $h^i_{a,b-1}$, and $h^i_{a-1,b}$;
   d) if c is even, d is odd, and $d < Y^i$, then retrieve the values $h^i_{c-1,d+1}$ and $h^i_{c,d+1}$;
   e) if c is odd, d is even, and $c < X^i$, then retrieve the values $h^i_{c+1,d-1}$ and $h^i_{c+1,d}$;
   f) if both c and d are odd, then:
      i) retrieve the value $h^i_{c+1,d}$ from H^q if $c < X^i$;
      ii) retrieve the value $h^i_{c,d+1}$ from H^q if $d < Y^i$;
      iii) retrieve the value $h^i_{c+1,d+1}$ from H^q if $c < X^i$ and $d < Y^i$.
3) Set a to be $\lceil a/2 \rceil$, b to be $\lceil b/2 \rceil$, c to be $\lceil c/2 \rceil$, and d to be $\lceil d/2 \rceil$.
4) Compute the next-level hash values as $h^{i+1}_{x,y} = \mathrm{H}(h^i_{2x-1,2y-1} \| h^i_{2x-1,2y} \| h^i_{2x,2y-1} \| h^i_{2x,2y})$ for $a \le x \le c$ and $b \le y \le d$.
5) Increment the value of i by 1, and go to Step 3 if i is smaller than $r = \max(\lceil \log_2 X \rceil, \lceil \log_2 Y \rceil) + 1$.
6) Verify the quotation by Verify $(h^r_{1,1}, g^r_{1,1})$ and output the result.

The above algorithm basically retrieves values from H^q where needed such that the next-level hash values can be calculated from the currently available hash values in a form of a rectangle with a top-left hash value $h^i_{a,b}$ and a bottom-right one $h^i_{c,d}$. The process continues until the tree root hash value $h^r_{1,1}$ is computed finally. Since $h^r_{1,1}$ is constructed by repeated applications of hashing followed by concatenation, it is difficult for an attacker to craft a forged quotation and a corresponding hash set that can ensure the same value $h^r_{1,1}$ to be constructed in the same way as that in the case of a legitimate quotation.

A 2-D TRUST source signature always contains just a single digital signature $g^r_{1,1}$, and its size is thus of the order $O(1)$. On the other hand, a 2-D TRUST complementary hash set H^q has no more than $6(r-1)$ hash values since at most three hash values are added to H^q for Steps 2a, 2b, and 2c; at most three hash values are added for Steps 2d, 2e, and 2f; and these steps are repeated for exactly $(r-1)$ times. Hence, the size of H^q, and also that of G^q, have an order of $O(\log_2 X + \log_2 Y)$, or equivalently, $O(\log_2 XY)$ because $r = \max(\lceil \log_2 X \rceil, \lceil \log_2 Y \rceil) + 1$. The total overhead size of the 2-D TRUST is thus $O(P \log_2 XY + Q \log_2 XY)$, contrasted with the originally larger total overhead size of $O(PXY + QXY)$ for the basic quote-the-whole technique.

C. Including Column and Row Headers

When a message author A^M quotes a two-dimensional rectangle of cells, the corresponding column and row headers usually need to be included as well. In the example illustrated by Fig. 2, the column headers specify the fiscal year of the quoted revenue figures ("2006", "2007", and "2008" in Fig. 2), while the row headers specify the types of income or expenditure ("Total Advertising Revenues" and "Licensing & other revenues" in Fig. 2). Both headers are critical and thus must be quoted along with the cells containing the actual revenue figures.

Since column and row headers are themselves one-dimensional, we can use the same techniques proposed in Sections III-C or D to quote the corresponding headers efficiently. That is, we can simply regard, for example, the row headers $s_{1,1}s_{1,2}, \ldots, s_{1,Y}$ as the sentences $s_1 s_2, \ldots, s_L$, and

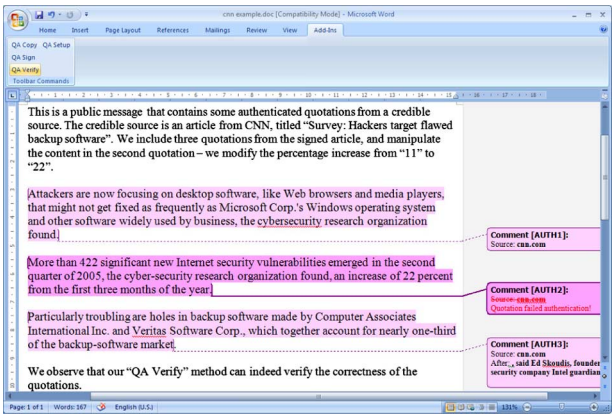


Fig. 3. Screenshot of Microsoft Word with the prototype add-in installed, which has added buttons in the toolbar for the purpose of quotation authentication.

use the MUST or TRUST to create suitable source and quotation signatures.

To ensure that the quoted rectangular range of cells matches the quoted column and row headers, we propose to include the column and row indexes of the respective cells when calculating their hash values. That is, instead of simply calculating $h^1_{x,y} = H(s_{x,y})$, we let $h^1_{x,y} = H(s_{x,y}||H(x)||H(y))$. This technique allows a message reader to verify the positions and associations of the quoted cells and headers. It can be figured out that if the signatures of the column and row headers are generated using the TRUST, the total overhead size remains to be of the order $O(P\log_2 XY + Q\log_2 XY)$.

V. Data Association Techniques for Quotation Authentication

In this section, the proposed data association techniques that are suitable for the popular Microsoft Word and Excel documents to allow for effective quotation authentication are described. Similar techniques for other media types can be used by applying appropriate data association techniques for those media [11]–[13].

A. Quotation Authentication Add-In

The Microsoft Word and Excel applications allow *add-ins* [14] to be installed to customarily expand their capabilities. A prototype quotation authentication add-in has been implemented in this study using the C# programming language, which installs several buttons on the toolbar of the Word (or Excel) application, as shown in Fig. 3. In this way, the three parties involved in the problem of quotation authentication can perform their respective tasks as simple as a click of a button.

In more detail, a source author generates a source signature for a source document by clicking on the "QA Sign" button of the add-in, which performs the following steps:

1) generate an appropriate source signature for the source document using the proposed technique;
2) transform the binary source signature into text by Base64 encoding; and
3) store the transformed signature in the *document properties* of the document.

It is convenient to embed the source signature directly into the document as described above so that the signature does not have to be published separately. This is possible because both Word and Excel documents allow arbitrary string values to be stored as key-value pairs in their special document properties dictionary. It is noted that the source signature generation process may be performed automatically whenever the source author saves a source document, thus relieving the need of manually invoking the "QA Sign" operation.

In this study, we assume that when a message author quotes a two-dimensional range of cells from a Microsoft Excel spreadsheet document, an appropriate authenticable quotation is generated and placed into a *Microsoft Word document as a table*, since this is the most common scenario. There are many large spreadsheets that contain information suitable for quoting, for example company financial statements, results of national voting, sales figures, and so on. A portion of such a large spreadsheet is often quoted and included as a table in a Word document. The processes of generating and verifying authenticable quotations from Word and Excel documents are mostly similar. We describe only the case of quoting from Word documents in detail in this study.

When a message author receives a source document, he/she can select any quotation in the document and click on the "QA Copy" button of the add-in to activate the authenticable quotation generation functionality, which performs the following steps:

1) extract and decode the source signature embedded in the document properties dictionary of the source document;
2) generate a quotation signature G^q for the selected quotation q using the proposed technique as described previously;
3) create an integrated authenticable quotation q' from q and G^q using the proposed data association techniques described in Section V-B; and
4) put q' into the system clipboard so that the message author can easily paste q' into his/her own document.

B. Integrated Authenticable Quotation

An authenticable quotation created as described above carries three different pieces of information. The first is the quotation itself and the second is the human-readable source author identity, both of which should be conveyed to the message reader directly. The third is an appropriate quotation signature that can be processed by the add-in to verify the fidelity and source of the quotation, and should ideally be invisible to the reader.

We propose to store the source author information in the *visible comments* section of a Word document, as shown in Fig. 3. Such comments are usually displayed at the right-hand side in Microsoft Word and clearly recognizable. A comment in Microsoft Word can be attached to any range of texts in the document, and we use this property to demarcate the span of a quotation. Also, each comment in a Word document has an *author* field to help distinguish different commentators during collaborative editing. This author field is utilized in this study to mark

comments that are used for quotation authentication by using a special value of "AUTH". Finally, if a message author quotes an incomplete sentence, we include the unquoted parts of the sentence in the comments as well for the reader's reference, and also to allow reconstruction of complete sentences for the purpose of quotation authentication.

We propose to embed the authentication information invisibly into an authenticable quotation by first transforming the binary data into normal text by Base 64 encoding to avoid misinterpretation. The transformed text is then inserted just after the first character of the quotation and made invisible by setting its "Font Effects" to "Hidden" [15]. It has been verified by experiments that this technique can be used to embed arbitrary long information invisibly into a Microsoft Word document.

The integrated authenticable quotation that contains the quotation itself, the visible comment, and the invisible authentication information as described is generated automatically by the add-in by a click of the button. The authenticable quotation is always copied in its entirety in copy-and-paste operations, making it easy for a message author to include such authenticable quotations when composing documents.

C. Identification and Verification of Authenticable Quotations

On receiving a document containing authenticable quotations, a message reader can verify the quotations by clicking on the "QA Verify" button installed by the authentication add-in, which performs the following quotation verification for each authenticable quotation:

1) scan the document for comments that contain the special author field "AUTH" and identify the span of a quotation by the range covered by the comment;
2) extract the hidden authentication information in the quotation and reversely transform the text version back to the binary original by Base64 decoding;
3) extract the partial sentences contained in the comment to reconstruct the full quotation sentences;
4) verify the fidelity of the quotation using the proposed quotation verification technique; and
5) format the comment of the quotation to show the result of quotation verification, as illustrated in Fig. 3.

VI. Conclusions and Future Work

The problem of quotation authentication has been described in this study, and a new approach to solving the problem has been proposed that allows message readers to efficiently verify quotations cited from known sources but embedded in messages by untrusted message authors. The proposed approach only requires the three parties involved in the problem to perform simple steps, without requiring a trusted third party to endorse quotations or requiring a message reader to access the original source article. Specifically, a source author is allowed to generate an appropriate source signature, such that any message author can generate a suitable quotation signature for arbitrary quotations from the source. The quotation signature is bundled together with the quotation using the proposed data association techniques to form an integrated authenticable quotation that can easily be copied and pasted to any message. Finally, a message reader can identify any authenticable quotations present in a message, and efficiently verify the source and the fidelity of the quotation.

We have described the basic enumerate-all-quotations and quote-the-whole techniques to demonstrate the feasibility of the proposed approach. A multiuse signature technique and a tree root uni-signature technique were then proposed to allow more efficient generation of source and quotation signatures. The two techniques have their respective merits depending on whether the message is widely distributed or not. The MUST is more efficient if there are a large number of message readers for a message, while the TRUST is better otherwise.

We also have pointed out the existence of nonsequential quotations, and in particular quotations that are two-dimensional rectangles of cells in spreadsheet documents. We have proposed a 2-D TRUST for efficient generation of suitable source and quotation signatures, such that the total overhead size is only of the order of the logarithm of the dimensions of the spreadsheet document.

Finally, specific data association techniques suitable for embedding source and quotation signatures in the Microsoft Word and Microsoft Excel documents have been proposed to demonstrate the feasibility of the proposed techniques. Furthermore, add-ins that can be installed in the Microsoft Word or Excel applications were described, which allow the three parties of the quotation authentication problem to perform their tasks easily.

In this paper, we proposed the 2-D TRUST as a better technique improving on the quote-the-whole technique for two-dimensional data. Better techniques improving on the enumerate-all-quotations technique is left as a possible future work. The problem of quotation authentication is an under-studied area of research, and studies of techniques suitable for other popular formats such as e-mail messages and Internet web pages, for example, are possible future works.

References

[1] S. Brain and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Comput. Netw. ISDN Syst.*, pp. 107–117, 1998.

[2] C. Fernstrom, "Management of trusted citations," in *Proc. 2003 ACM Symp. Document Engineering*, Grenoble, France, 2003, pp. 243–245.

[3] M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash functions for message authentication," in *Advances in Cryptology–Crypto '96*, 1996, LNCS 1109, pp. 1–15.

[4] H. Krawczyk, M. Bellare, and R. Canetti, HMAC: Keyed-hashing for message authentication RFC 2104, Feb. 1997.

[5] W. Stallings, *Cryptography and Network Security, Principles and Practice*, Second ed. Englewood Cliffs, NJ: Prentice-Hall Int., 1999.

[6] *RSA Cryptography Standard*, PKCS #1 V2.1, RSA Laboratories, Jun. 14, 2002.

[7] S. Josefsson, Storing Certificates in the Domain Name System (DNS) RFC 4398, Mar. 2006 [Online]. Available: http://www.ietf.org/rfc/rfc4398.txt

[8] R. Gennaro and P. Rohatgi, "How to sign digital streams," *Inf. Computation Archive*, vol. 165, no. 1, pp. 100–116, 2001.

[9] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, "Efficient and secure source authentication for multicast," in *Proc. Network and Distributed System Security Symp.*, 2001, pp. 35–46.

[10] X. Wang, Y. L. Yin, and H. Yu, "Finding collisions in the full SHA-1," *Lecture Notes Comput. Sci.*, vol. 3621, pp. 17–36, 2005.

[11] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information hiding—a survey," *Proc. IEEE*, vol. 87, pp. 1062–1078, 1999.

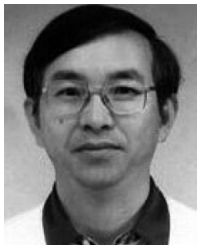
[12] N. F. Johnson, Z. Duric, and S. Jajodia, *Information Hiding: Steganography and Watermarking—Attacks and Countermeasures*. Boston, MA: Kluwer, 2001.

[13] I. J. Cox and M. L. Miller, "The first 50 years of electronic watermarking," *J. Appl. Signal Process.*, vol. 2, pp. 126–132, 2002.

[14] F. C. Rice, "Building a COM add-in for Microsoft Office XP using Microsoft Visual Basic 6.0," *Microsoft Office XP Technical Articles*, 2002, Microsoft Corporation.
[15] H. Carvey, *Windows Forensics and Incident Recovery*. Boston, MA, USA: Addison-Wesley, 2004.

Tsung-Yuan Liu (S'04) received the B.S. degree in electrical engineering from the University of the Witwatersrand, Johannesburg, South Africa, the M.B.A. degree from National Taiwan University, Taipei, Taiwan, and is currently pursuing the Ph. D. degree at the College of Computer Science, National Chiao Tung University, Hsinchu, Taiwan.

He is a Software Engineer with Google, Taipei, Taiwan. His research interests include information hiding, image processing, web search, data mining, and artificial intelligence.



Wen-Hsiang Tsai (S'78–M'79–SM'91) received the B. S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, the M.S. degree from Brown University, Providence, RI, and the Ph.D. degree from Purdue University, West Lafayette, IN.

Currently, he is a Chair Professor with the Department of Computer Science, National Chiao-Tung University, Hsinchu, Taiwan, and was the President of Asia University, Taichung, Taiwan. So far he has published 142 journal papers and 226 conference papers. His research interests include image processing, computer vision, information security, and autonomous vehicle applications.