CERIAS Tech Report 2005-131
Indexing Information for Data Forensics
by Mikhail J. Atallah
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

Indexing Information for Data Forensics

Michael T. Goodrich[1], Mikhail J. Atallah[2], and Roberto Tamassia[3]

[1] University of California, Irvine. goodrich(at)uci.edu
[2] Purdue University. mja(at)cs.purdue.edu
[3] Brown University. rt(at)cs.brown.edu

Abstract. We introduce novel techniques for organizing the indexing structures of how data is stored so that alterations from an original version can be detected and the changed values specifically identified. We give forensic constructions for several fundamental data structures, including arrays, linked lists, binary search trees, skip lists, and hash tables. Some of our constructions are based on a new reduced-randomness construction for nonadaptive combinatorial group testing.

Keywords: data forensics, data integrity, data marking, combinatorial group testing, information hiding, tamper detection, data structures

1 Introduction

Computer forensics [71] deals with methods for extracting digital evidence after a computer crime has been committed. Typically, such crimes involve modifying documents, databases, and other data structures to an attacker's advantage. Examples could include a student changing a grade in a registrar's database, a dishonest speculator altering online financial data for a certain company, an identity thief modifying personal information of a victim, or a computer intruder altering system logs to mask a virus infection. It would be ideal in such cases if an investigator could identify, after the fact, which pieces of information were changed and, in so doing, be able to implicate the attacker. In the rest of this section, we describe our motivation, model, and related work, and we summarize our contributions. But before doing so, we briefly give a simplified and intuitive overview of what this paper is about.

A cryptographic one-way hash is a commonly used way of detecting unauthorized or otherwise malicious modification of a file or other digital object (e.g., [5, 44, 62], to mention a few of many examples). This is done by storing a keyed cryptographic hash of the item and using it later as a reference for comparison. This paper is about going beyond the yes/no afforded by this common use of cryptographic hashes: given n items, we now seek to store as few hashes as possible so as to enable the pinpointing of *which* of these n items were modified (by comparing the computed hashes to the stored hash values). Of course a hash is now applied to (a concatenation of) a subset of the n items. But *which* subsets, and *how many* of them, are needed so as to pinpoint the modifications of up to d of the n items ? We show that remarkably few hashes suffice. Why it is so important to use few hashes will become apparent when we consider the application we describe for the above-mentioned combinatorial result: the case when the n items are in the nodes of a data structure, and we seek to store the hashes within the topology of the data structure, i.e., *without using any additional space* (and, of course, without modifying any of the n items stored in the data structure). In other words, this forensic marking comes “for free” as far as the space of the data structure is concerned.

Motivation. As mentioned above, in this paper we initiate an investigation into methods for encoding information in the way data is indexed so as to identify if it has been altered, and if so, to determine exactly the pieces of information that have changed. Formally, we model this problem in terms of a data structure D that is stored on a semi-trusted machine, which under normal circumstances, would use D for some desired purpose. If there is an indication or suspicion that D has been altered in a malicious way, then we would like to enable a computer investigator, whom we call the *auditor*, to examine the current state of D to determine what, if anything, has changed. Of course, a trivial solution would be for the auditor to cache a copy of D in protected memory and do a simple comparison of this copy of D to the current version. This solution would achieve the desired goal of identifying exactly the parts of D that have changed, but it would also require a tremendous amount of storage for the auditor, who is potentially responsible for a large community of users and computers. Thus, we additionally restrict solutions to use no storage at the auditor (or equivalently in protected memories), other than possibly some small number of administrative values, such as a master key for “unlocking” information encoded in D. We refer to this problem as that of *indexing information for data forensics*, or *data forensics marking*, for short.

Model. We are further interested only in solutions where marking the indexing structure of data leaves the actual data values unchanged, since changing data values could alter the outcome of queries in unintended ways. Unlike well-known digital watermarking techniques [18], we want to encode authentication information only in the organizational structure of a data structure D, not in the values stored in it. We allow ourselves the possibility of modifying non-data fields in D, but we require that any such changes we carry out be stealthy, that is, not immediately detectable by the adversary. In information-hiding terminology, we view the information hidden in D to be steganographic rather than a watermark (which, strictly speaking, does not require stealthiness, whereas steganography requires that even the presence of hidden information be undetectable). Ideally, we want to encode information only in the topology of D’s pointers and the ordering of D’s memory blocks, yet we desire there to be sufficient information so as to specifically identify any portions of D that have been changed by an attacker. Note that this requirement for pinpointing of the changes goes beyond the notion of making the structure tamper-evident in the usual yes/no sense, something usually achieved using HMACs and digital signatures, but whose yes/no outcome has too coarse a granularity for our purposes.

Our model of the adversary is as follows. The adversary has access to the data structure D after it has been deployed on the semi-trusted machine and can modify the values of D, but not the topology of D’s pointers and the ordering of D’s memory blocks. This assumption is realistic in many practical applications for the following two reasons. First, regular users typically provide data to an application through the application’s user interface but cannot modify the application’s internal storage or the application's code to alter the memory arrangement of the data. Second, even if the user were able to alter the data organization, the user may not think about doing it because of the stealthiness of the marking.

The adversary is successful if he can modify some values of D without such changes being detected by the auditor. The adversary has knowledge of the algorithm the auditor will use to perform a forensic analysis of D. However, we do not allow the adversary to know the cryptographic master key maintained by the auditor, nor of any keys that are derived from such a master key. This is the usual (and preferred) "white box" security requirement. Although in practice one may gain additional "security through obscurity", it is wiser to assume the adversary knows all but the keys. We illustrate this model in Figure 1.

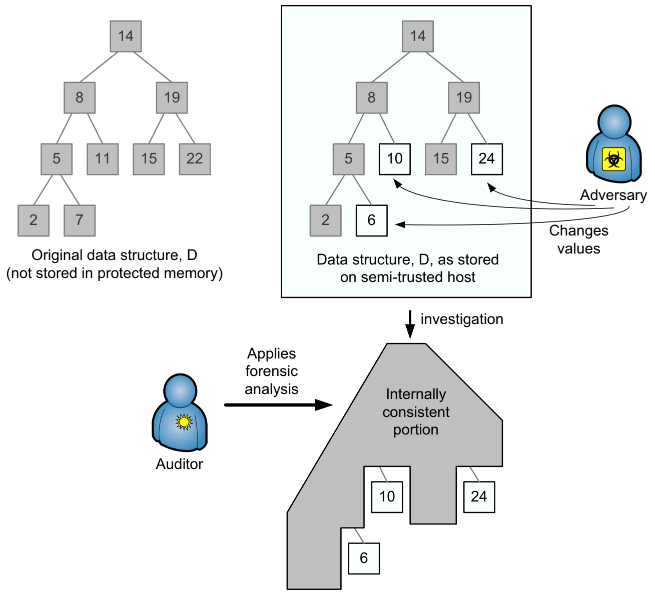

Fig. 1. An illustration of data forensics marking.

Prior Related Work. Computer forensics has been applied to software authorship detection [43, 63], the integrity of audit logs using external protected memories [59], tracing IP packets during denial-of-service attacks [29, 57], and e-mail author identification [20]. We are not familiar with any previous work on data forensics marking, however. Nevertheless, there has been considerable prior work in the theoretical computer science literature on a number of related areas, including digital watermarking, combinatorial group testing, program checking, property testing, and authenticated data structures. We review some of this prior work here.

Digital watermarking [18] deals with methods for hiding a mark (usually identifying information) in digital content in a manner that is resilient, i.e., hard to remove by an adversary without considerably damaging the object. There are many applications of watermarking, including inserting ownership information, inserting purchaser information, placing caption information, etc. Most watermarking work has been on multimedia content, where minor degradation of the quality of the media is acceptable in order to make the mark more resilient. There have been some notable exceptions, dealing with watermarking software, semi-structures (XML), and relational databases. Collberg and Thomborson [19, 17], Chang and Atallah [15], Horne, Matheson, Sheehan, and Tarjan [38] and Venkatesan, Vazirani, and Sinha [70], present schemes for watermarking software. Qu and Potkonjak [55] propose a watermarking scheme for graph colorings. Khanna and Zane [39] describe a scheme for encoding information in the weights of a graph representing a map so as to preserve shortest paths. Gross-Amblard [35] studies ways of changing values to encode identifying information in a database or XML document so as to preserve the answers to certain queries. Marking XML structures is also the topic of Sion et al. [60]. Database watermarking that slightly degrades the data was proposed by Agarwal et al. [1] and Sion et al. [61]. This watermarking work is related to data forensics marking in that it is directed at encoding information in digital content. It differs from data forensics marking, however, in that digital watermarking allows data values to change (in hopefully imperceptible ways) and is not interested in identifying the specific places where content has changed.

Several researchers have studied combinatorial group testing and its applications to cryptography and information encoding (e.g., see [16, 26]). This area is directed at performing group tests on subsets of a given set S to identify defective elements in S. The area has not to date been applied to data index integrity, but in this paper we show an interesting connection between data forensics marking and a new reduced-randomness construction of a nonadaptive combinatorial group testing scheme, which may be of independent interest. As evidence for this claim, we observe that Kurosawa, Johansson, and Stinson [45] explore other applications of reduced-randomness constructions in cryptography, and Stinson, van Trung, and Wei [65] explore applications of group testing to key distribution in cryptography.

Following early work on program checking (see, e.g., [7, 66, 67]), efficient schemes have been developed for checking the results of various data structures and algorithms (see, e.g., [4, 8–10, 23, 24, 27, 40, 48, 49]). These schemes typically utilize linear space with checking algorithms that run faster than the construction algorithms they are checking, and they are directed at detecting if an algorithm has performed correctly or not. We are not aware, however, of any prior work on program checking that, in addition to detecting an error state, also identifies all the places in a program or structure that have become invalid. Likewise, the related area of property testing is directed at determining if a combinatorial structure satisfies a certain property or is "far" from such a structure (e.g., see [56]), and it too does not identify all property violations.

Prior work in the area of authenticated data structures [3, 11, 13, 14, 21, 22, 28, 30–34, 41, 42, 47, 52, 54, 68, 69] has focused on disseminating information from a single, trusted source so that an alteration to the data structure could be detected, but not specifically identified. That is, they do not provide solutions to the data forensics marking problem. Related work on committed databases has recently been presented in [53].

Our Contributions. In this paper, we introduce the data forensics marking framework and give several results for this model that use no additional storage at the auditor other than a master key. The security of our methods are based on standard cryptographic assumptions. Namely, we assume the existence of the *message authentication code* (*MAC*) cryptographic primitive, which is a key-dependent one-way collision-resistant function (see, e.g., [50, 58, 64]). A message authentication code can be constructed from a standard cryptographic hash function.

We give forensic constructions for several fundamental data structures, including arrays, linked lists, binary search trees, skip lists, and hash tables. Some of our constructions are based on a new reduced-randomness construction for nonadaptive combinatorial group testing, which is of independent interest.

All of our data forensics marking constructions involve two phases of computation. In the first phase, we build a program P and authentication information A for S so that P can detect and identify up to some number, d, of changes to S in the indexing structure D using A. In the second phase, we encode P and A in the organizational and topological structure of D in a way that is probabilistically difficult for the adversary to reproduce, yet it also preserves the accuracy of P with high probability or it still allows P to restrict the changed values in S to a small set of candidates (which is often sufficient for forensics). The challenge, of course, is to design the encoding of P and A in D so that it can survive up to d alterations of the values in S, as stored in D, that the adversary might make.

Our contributions can be summarized as follows:

– We develop a new reduced-randomness construction for nonadaptive combinatorial group testing. In particular, we show how to construct a $t \times n$ d-disjunct binary matrix M encoding a nonadaptive combinatorial group test for n items that can detect up to d defective items, with t being $O(d^2 \log n)$. Our construction uses only $O(d^3 \log n \log d)$ random bits and is correct with high probability, whereas previous schemes use $\Theta(d^2 n \log n)$ random bits and are not high-probability constructions. Thus, we can encode matrix M using $O(\log n)$ bits when d is a constant, a polylogarithmic number of bits when d is polylogarithmic, or $o(n)$ bits when d is $o(n^{1/3}/\log^{2/3} n)$, which is of independent interest.
– We give efficient forensic constructions of several fundamental data structures, including binary search trees, skip lists, arrays, linked lists, and hash tables. The number of changes we can detect and identify for a data structure D storing a set S of n elements is $O(n^{1/3}/\log^{2/3} n)$ for balanced search trees and skip lists, $O(n^{1/4}/\log^{1/4} n)$ for arrays and linked lists, and $O(1)$ for hash tables.

In the next section, we describe our reduced-randomness construction of a nonadaptive combinatorial group test. In Section 3, we outline our two-phase constructions of indexing structures for data forensics marking schemes. We conclude in Section 4.

2 Blood Testing and Forensics

As mentioned above, some of our solutions utilize a reduced-randomness construction of a nonadaptive combinatorial group testing scheme. Combinatorial group testing [26] (or "CGT") schemes identify "bad" members of a set S of n elements using group tests. A *group test* consists of selecting a test sample $T \subset S$ and performing a single experiment to determine whether or not T contains a bad element. A testing scheme that makes all its tests in a single round, with all test sets determined in advance, is said to be *nonadaptive*. Most efficient schemes are designed assuming there is an upper bound, d, on the number of possible bad members of the input set S, where $1 \leq d < n$. Combinatorial group testing was originally formulated for testing blood supplies during World War II, with a group test comprising a tester extracting a few drops from each blood sample in a test set, pooling them together, and testing the mixed sample for the syphilis antigen [25].

Reduced-Randomness Nonadaptive Combinatorial Group Testing. For the case $d = 1$, it is straightforward to design a nonadaptive scheme using $O(\log n)$ tests. For the general case, $d > 1$, however, designing efficient general testing schemes is more challenging. Adaptive schemes generally make fewer total tests, in terms of d and n, than nonadaptive schemes. In particular, the best known general-purpose adaptive schemes use $O(d \log(n/d))$ tests, whereas the number of tests used by the best known general-purpose nonadaptive schemes is $O(d^2 \log n)$ [26]. Even so, adaptive schemes are not applicable in many contexts, including DNA sequence analysis and the context of this paper.

Our application of combinatorial group testing to data forensics marking is based on the use of nonadaptive CGT schemes. Unfortunately, the known deterministic nonadaptive CGT schemes are asymptotically suboptimal or not designed for most values of d and n, and the known randomized CGT schemes, for $d \geq 2$, utilize $\Theta(d^2 n \log n)$ random bits (e.g., see [26]). These drawbacks make existing nonadaptive combinatorial group testing schemes infeasible for data forensics marking, where we wish to limit the memory requirements of the auditor.

In this section, we present a simple, randomized nonadaptive combinatorial group testing scheme, for $d \geq 2$, where we reduce the needed random bits to be $O(d^3 \log n \log d)$. This reduced-randomness CGT scheme is based on applying the construction of Alon *et al.* [2] of almost k-wise independent random variables (see also [6, 51]) to the randomized CGT approach of Busschbach [12], and then showing that almost k-wise independent random variables can be used to achieve an efficient nonadaptive CGT with high probability (which is, in fact, a stronger result than the previous algorithm achieves using fully independent variables). The main idea of this approach is to construct a $t \times n$ binary matrix M, where each column corresponds to an element of S and each row corresponds to a test—so that $M[i, j] = 1$ if and only if element j is included in test i.

A $t \times n$ binary matrix M is *d-disjunct* [26] if, for any $d + 1$ columns with one of them designated, there always exists a row with a 1 in the designated column and 0's in the other d columns. Given a d-disjunct binary matrix M, we can immediately design a nonadaptive combinatorial group testing scheme—simply perform the test indicated by each row of M. With the results of these tests in hand, we can then remove each column of M that has a 1 in a row that returned a negative test result (recall that, in the testing framework, a negative is a good outcome). The remaining columns correspond to the "bad" elements. The correctness of this algorithm is derived directly from M being d-disjunct, for if we designate a "good" column together with a group of up to d bad columns, there must be a row (i.e., a test) that includes the good column and excludes the bad ones. See Figure 2.

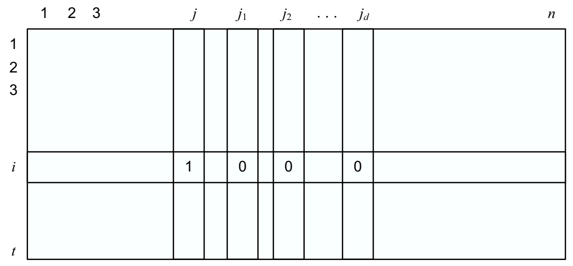

Fig. 2. An illustration of a $t \times n$ d-disjunct matrix.

Our algorithm for building a d-disjunct $t \times n$ matrix M is simply to set each $M[i,j] = 1$ with probability roughly $\frac{1}{d+1}$, by using "almost" k-wise independent random variables. The notion of being almost k-wise independent that we use is based on the following definition of Alon *et al.* [2]:

Definition 1. *A set of probability-$(1/2)$ random bits, $x_1, x_2, \ldots, x_n$, is (ϵ, k)-independent if, for any k-bit vector α and k positions $i_1 < i_2 < \cdots < i_k$, we have*

$$|\Pr[x_{i_1} x_{i_2} \cdots x_{i_k} = \alpha] - 2^{-k}| \leq \epsilon.$$

Alon *et al.* [2] establish the following important fact:

Lemma 1. *Let $N = 2^r - 1$ and let k be an odd integer. Then, using*

$$2\left(\left\lceil \log \frac{1}{\epsilon} + \log\left(1 + \frac{(k-1)r}{2}\right)\right\rceil\right)$$

probability-$(1/2)$ random bits, one can construct a set of N probability-$(1/2)$ bits that are (ϵ, k)-independent. That is, the number of needed random bits is roughly $2\log\left(\frac{k \log N}{2\epsilon}\right)$.

Given Lemma 1, we would like to set t so that the $t \times n$ random matrix M we construct will be d-disjunct with high probability.

Theorem 1. *Given integers d and n such that $2 \leq d < n$, one can construct a $t \times n$ binary matrix M that is d-disjunct with high probability, using $O(d^3 \log n \log d)$ random bits, where t is $\Theta(d^2 \log n)$.*

Proof. Let R be a set of tnl probability-$(1/2)$ random bits that are $(\epsilon, (d+1)tl)$-independent, where $l = \lceil \log(d+1) \rceil$ and t and ϵ will be set in the analysis. Considering these bits l at a time, we can convert this set into a set R' of probability-p bits, where $p = 1/2^l$, so that $1/2(d+1) < p \leq 1/(d+1)$. We use the bits in R' to define the matrix M so that $M[i,j] = 1$ if and only if the corresponding bit in R' is equal to 1. (Note:

Azar, Motwani, and Naor [6] have an alternate, more general, approach for constructing almost k-independent probability-p bits, but their construction is more complicated than what is needed here.)

Consider now a particular column j and d other columns $j_1, j_2, \ldots, j_d$ in matrix M. For any row i in M, had the random variables in R' been at least $(d+1)$-wise independent, then the probability that $M[i,j] = 1$ and $M[i,j_s] = 0$, for $s = 1, 2, \ldots, d$, would be $p(1-p)^d$. Thus, had the variables in R' been at least $(d+1)t$-wise independent, then the probability that no such row exists (a *failure*) among these columns would be

$$\left[1 - p(1-p)^d\right]^t.$$

Notice that this probability is actually determined by $(d+1)tl$ bits in R. Let $\mathcal{F}$ denote the set of all vectors of values for these $k = (d+1)tl$ bits $x_{i_1}x_{i_2}\ldots x_{i_k}$ that result in a failure event for column j and the d other columns, and note that $|\mathcal{F}| \le 2^k$. Then, by Lemma 1, for each vector α in $\mathcal{F}$, we have

$$|\Pr[x_{i_1}x_{i_2}\cdots x_{i_k} = \alpha] - 2^{-k}| \le \epsilon.$$

That is, we obtain

$$2^{-k} - \epsilon \le \Pr[x_{i_1}x_{i_2}\cdots x_{i_k} = \alpha] \le 2^{-k} + \epsilon.$$

In other words, this probability is bounded from above by ϵ plus the value it would have been had these bits been k-wise independent. Therefore, we have

$$\sum_{\alpha\in\mathcal{F}} \Pr[x_{i_1}x_{i_2}\cdots x_{i_k} = \alpha] \;\le\; \left[1 - p(1-p)^d\right]^t + 2^k\epsilon.$$

There are $(d+1)\binom{n}{d+1}$ ways of distinguishing a column j and d other columns in M. Moreover, the probability that any column j and d others determine a failure is certainly no more than the probability that all such groups determine a failure, which, irrespective of any considerations about the independence of the underlying random variables, is no more than

$$(d+1)\binom{n}{d+1}\left[1 - p(1-p)^d\right]^t + (d+1)\binom{n}{d+1}2^k\epsilon.$$

By the definition of p, this probability is at most

$$(d+1)\binom{n}{d+1}\left[1 - \frac{1}{2(d+1)}\left(1 - \frac{1}{d+1}\right)^d\right]^t + (d+1)\binom{n}{d+1}2^k\epsilon.$$

Using the inequalities $(1 - 1/(d+1))^d > 1/3$ and $(d+1)\binom{n}{d+1} \le n^{d+1}$, for $d \ge 2$, and substituting in the value for k, we can further simplify this probability as being at most

$$n^{d+1}\left(1 - \frac{1}{6(d+1)}\right)^t + n^{d+1}2^{(d+1)tl}\epsilon.$$

For our claim to hold with high probability, we would like each of the above terms to be at most $1/n$. To bound the first term by $1/n$, we can use the inequality $-\ln(1-x) \geq x$, for $0 \leq x < 1$, and we can set

$$t = 6(d+1)(d+2)\lceil \ln n \rceil,$$

which is $\Theta(d^2 \log n)$. Given this value for t, to then bound the second term by $1/n$, we can set

$$\epsilon = n^{-(d+2)} 2^{-6(d+1)^2(d+2)\lceil \ln n \rceil \lceil \log d \rceil}.$$

According to Lemma 1, the number of random bits needed for this construction is

$$2\left(\left\lceil \log \frac{1}{\epsilon} + \log\left(1 + \frac{(k-1)\log N}{2}\right)\right\rceil\right).$$

That is, the number of random bits needed is $O(d^3 \log n \log d)$.

Having a reduced-randomness construction, as specified in Theorem 1, allows us to encode a $t \times n$ d-disjunct binary matrix M simply by storing the $O(d^3 \log n \log d)$ random bits used to generate M.

3 Specific Constructions for Data Forensics Marking

We use Theorem 1 in many of our data forensics marking solutions, which we briefly outline in this section. Throughout this discussion, we assume the reader is familiar with the fundamental data structures mentioned. In addition, throughout this section we assume that the auditor and data structure designer share a secret key K that is easily derived from the auditor's master key and the data structure designer's identity. We also assume the existence of a message authentication code (MAC) function $f_K(x)$ with key K. (see, e.g., [50, 58, 64]).

From Test Samples to Set Integrity Checking. Given a nonadaptive combinatorial group test (CGT) for detecting up to d defectives in a set of n items, we can convert this into a test of the integrity of a collection S of n items stored in a data structure. For each test T specified by the CGT, compute its *authentication value* a_T defined by

$$a_T = f_K(x_1 \,||\, x_2 \,||\, \cdots \,||\, x_m),$$

where f_K is a MAC function and $x_1, \cdots x_m$ are the items of S included in test T, in sorted order. We can then recompute a_T on an altered copy of S to determine if this value has changed. If so, then we know that an item in T has been modified. Thus, performing all these comparisons for all the tests in the CGT would give us a determination of which items in S have changed.

Lemma 2. *We can construct a forensic scheme for a set of n elements so as to detect and identify which of up to d of its elements have been changed, with high probability, using $O(d^3 \log^2 n)$ bits of authentication information.*

The problem, of course, is that the auditor does not have enough memory to store an encoding of a CGT. Nevertheless, Theorem 1 gives a way of avoiding storing anything at the auditor, for it implies that we can encode a nonadaptive CGT for detecting d defective elements among n using only $O(d^3 \log n \log d)$ bits. The number of tests determined by these bits is $O(d^2 \log n)$. For each test T defined by this CGT, we need to store the authentication value a_T, Note that a_T is determined by a message authentication code based on the key K, which is unknown to the adversary. Thus, the CGT and its associated authentication values can be represented using $O(d^3 \log^2 n)$ bits. Our solution, then, to avoid any storage at the auditor, is to encode these bits in the data structure itself.

Balanced Binary Search Trees. A balanced binary search tree holding n items has $O(\log n)$ depth. Its structure can also easily encode $O(n)$ bits in a simple recursive fashion. Given a set S of comparable items, we have $n/2$ items that we can pick for the root's value (from the middle half of elements from rank $n/4$ to $3n/4$). The exact choice allows us to encode $\log(n/2)$ bits at the root level, and we can then repeat this construction recursively at each child. Since we can encode in the tree a total of $O(n)$ bits we picking d that is $o(n^{1/3}/\log^{2/3} n)$.

By Lemma 2, we obtain the following result:

Theorem 2. *We can construct a forensic scheme for a balanced binary tree storing n elements so as to detect and identify which of up to $O(n^{1/3}/\log^{2/3} n)$ of its values have been changed, with high probability.*

Skip Lists. We can use, for skip lists, a scheme similar to the one developed for binary search trees. At each level of the skip list structure, about half of the (say) m elements of that level will survive to the next (higher) level. This allows us to encode at that level a number of bits equal to

$$\log \binom{m}{m/2} \geq \log(2^{m/2}) = m/2,$$

Hence, the overall encoding capacity in an n-element skip list is $O(n)$ bits. Thus, just as for the balanced binary tree case, by Lemma 2, we obtain the following result:

Theorem 3. *We can construct a forensic scheme for a skip list storing n elements so as to detect and identify which of up to $O(n^{1/3}/\log^{2/3} n)$ of its values have been changed, with high probability.*

Arrays and Linked Lists. Arrays and linked lists allow up to $O(n \log n)$ bits to be encoded in the permutation of the items stored in the list or array. But this information is stored implicitly, since the ordering itself could be altered should the adversary change values. Our solution in this case is to replicate the CGT and its expected values $d + 1$ times and spread these multiple encodings evenly across the bits encoded by the permutation. By a pigeon-hole argument, even if the adversary changes d values, there will still be a large enough contiguous run of unchanged values that encode the CGT and its expected values so as to allow us to determine which values might have changed.

In this case we need $O(d^4 \log^2 n)$ bits. Therefore, by Lemma 2, we have the following result:

Theorem 4. *We can construct a forensic scheme for an array or linked list storing n elements so as to detect and identify which of up to $O(n^{1/4}/\log^{1/4} n)$ of its values have been changed, with high probability.*

This construction can also be applied to relational databases that use a data-independent index number or unique ID to name records. In this case we can treat the index number in the same way we used positions in an array or linked list.

Hash Tables. A hash table for a set S of n elements consists of a bucket array B of size $O(n)$ and a hash function h which maps elements of S to cells of B under some collision-handling rule. Let us assume that h is based on a simple, standard linear function, so that $h(x) = \alpha x + \beta \bmod p$, for each x in S, where p is a prime on the order of n. There is not a lot of variability in such a hash table that we can exploit for forensic analysis, but there is nevertheless enough for the following:

Theorem 5. *A hash table has a forensic construction that can detect and identify a single value insertion or deletion and can isolate a changed value to a set of constant expected size.*

Proof. Our construction begins by choosing p to be a random prime on the order of n (for hash table efficiency, it is good that p is slightly larger than n) such that $p \bmod 4 = n \bmod 4$. Sort the elements of S and compute value α defined as follows:

$$\alpha = f_K(x_1 \,||\, x_2 \,||\, \cdots \,||\, x_n \,||\, p),$$

where $x_1, \cdots x_n$ are the items of S in sorted order. We then compute value β as follows:

$$\beta = x_1 \oplus x_2 \oplus \cdots \oplus f_0(p).$$

We then define the hash function h for the hash table as $h(x) = \alpha x + \beta \bmod p$.

Let us consider the forensic capabilities of this structure. If an item is deleted, then we can detect this case using α and we can recompute the deleted value from the XOR of β and the remaining values in the hash table. If a value is added, then we can detect this case using α and we can compute the complement of this value from the XOR of β and the existing values in the hash table. If a value is changed, then we can detect this case using α and β. For each element x in the hash table, we can use β to determine what its value should have been were x the item that changed, and then recompute α to verify that this is the case. The expected number of values that will be determined to have possibly changed will be $O(1)$ and the adversary cannot control this value, since he does not know K and is assumed to be unable to invert f.

Security. The security of our constructions is based on the fact that a successful adversary will have to invert or find collisions in the message authentication function f_K used or recover the secret key K shared by the data structure designer and auditor. A complete security proof will be given in the full version of the paper.

Extension. We can extend our results to a stronger adversarial model, where the adversary can also modify a constant fraction of the authentication information hidden in the data structure. For this purpose, we encode the authentication information using the cryptographic variation of the Guruswami-Sudan list decoder [36, 37] presented in [46]. Even with this stronger model, we obtain results analogous to those given in Theorems 2–4. Details will be given in the full version of the paper.

4 Conclusion

We have introduced the topic of information indexing for data forensics marking, given a new reduced-randomness construction for nonadaptive combinatorial group testing, and applied this and other techniques to design efficient and robust constructions of forensic schemes for several kinds of data indexing structures. We believe there is still a considerable amount of additional work that could be done in this area. In particular, it would be useful to have efficient forensic schemes for data that is changing over time. Such a solution would solve the problem of maintaining forensic data for audit logs and dynamic databases.

Acknowledgements

The work of the first author was supported in part by NSF Grants IIS-0325345, IIS-0219560, IIS-0312357, and IIS-0242421, ONR Contract N00014-02-1-0364, by sponsors of the Center for Education and Research in Information Assurance and Security, and by Purdue Discovery Park's e-Enterprise Center. The work of the second author was supported in part by NSF Grants CCR-0312760, CCR-0311720, CCR-0225642, and CCR-0098068. The work of the third author was supported in part by NSF Grants CCR-0311510, IIS-0324846 and CNS-0303577 and by a research gift from Sun Microsystems.

References

1. R. Agrawal and J. Kiernan. Watermarking relational databases. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Hong Kong*, pages 155–166. ACM Press, 2002.
2. N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple construction of almost k-wise independent random variables. *Random Structures and Algorithms*, 3:289–304, 1992.
3. A. Anagnostopoulos, M. T. Goodrich, and R. Tamassia. Persistent authenticated dictionaries and their applications. In *Proc. Information Security Conference (ISC 2001)*, volume 2200 of *LNCS*, pages 379–393. Springer-Verlag, 2001.
4. S. Ar, M. Blum, B. Codenotti, and P. Gemmell. Checking approximate computations over the reals. In *Proc. ACM Symp. on the Theory of Computing*, pages 786–795, 1993.
5. W. Arbaugh, D. Farber, and J. Smith. A secure and reliable bootstrap architecture, 1997.
6. Y. Azar, R. Motwani, and J. Naor. Approximating probability distributions using small sample spaces. *Combinatorica*, 18(2):151–171, 1998.
7. M. Blum and S. Kannan. Designing programs that check their work. *J. ACM*, 42(1):269–291, Jan. 1995.

8. J. D. Bright and G. Sullivan. Checking mergeable priority queues. In *Digest of the 24th Symposium on Fault-Tolerant Computing*, pages 144–153. IEEE Computer Society Press, 1994.
9. J. D. Bright and G. Sullivan. On-line error monitoring for several data structures. In *Digest of the 25th Symposium on Fault-Tolerant Computing*, pages 392–401. IEEE Computer Society Press, 1995.
10. J. D. Bright, G. Sullivan, and G. M. Masson. Checking the integrity of trees. In *Digest of the 25th Symposium on Fault-Tolerant Computing*, pages 402–411. IEEE Computer Society Press, 1995.
11. A. Buldas, P. Laud, and H. Lipmaa. Eliminating counterevidence with applications to accountable certificate management. *Journal of Computer Security*, 10(3):273–296, 2002.
12. P. Busschbach. Constructive methods to solve the problems of: s-sujectivity conflict resoltuion, coding in defective memories. Unpublished manuscript, cited in [26], 1984.
13. J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In M. Yung, editor, *Advances in Cryptology — CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 61–76. Springer Verlag, 2002.
14. S. Cannella, M. Shin, C. Straub, R. Tamassia, and D. J. Polivy. Secure visualization of authentication information: A case study. In *Proc. IEEE Symp. on Visual Languages and Human-Centric Computing*, 2004.
15. H. Chang and M. Atallah. Protecting software code by guards. In T. Sander, editor, *Security and Privacy in Digital Rights Management*, volume 2320 of *Lecture Notes in Computer Science*, pages 160–175. Springer-Verlag, 2002.
16. C. J. Colbourn, J. H. Dinitz, and D. R. Stinson. Applications of combinatorial designs to communications, cryptography, and networking. In Walker, editor, *Surveys in Combinatorics*, volume 187 of *London Mathematical Society Lecture Note Series*, pages 37–100. Cambridge University Press, 1993.
17. C. Collberg and C. Thomborson. On the limits of software watermarking. Technical Report 164, Department of Computer Science, The University of Auckland, Private Bag 92019, Auckland, New Zealand, Aug. 1998.
18. C. Collberg and C. Thomborson. Software watermarking: Models and dynamic embeddings. In *ACM Symp. on Principles of Programming Languages (POPL)*, pages 311–324, 1999.
19. C. Collberg and C. Thomborson. Software watermarking: models and dynamic embeddings. In *ACM SIGPLAN–SIGACT POPL'99*, San Antonio, Texas, USA, Jan. 1999.
20. O. de Vel, A. Anderson, M. Corney, and G. Mohay. Mining e-mail content for author identification forensics. *SIGMOD Record*, 30(4):55–64, 2001.
21. P. Devanbu, M. Gertz, A. Kwong, C. Martel, G. Nuckolls, and S. G. Stubblebine. Flexible authentication of XML documents. In *Proc. ACM Conf. on Computer and Communications Security*, pages 136–145, 2001.
22. P. Devanbu, M. Gertz, C. Martel, and S. G. Stubblebine. Authentic data publication over the internet. *Journal of Computer Security*, 11(3):291 – 314, 2003.
23. O. Devillers, G. Liotta, F. P. Preparata, and R. Tamassia. Checking the convexity of polytopes and the planarity of subdivisions. *Comput. Geom. Theory Appl.*, 11:187–208, 1998.
24. G. Di Battista and G. Liotta. Upward planarity checking: "Faces are more than polygons". In S. H. Whitesides, editor, *Graph Drawing (Proc. GD '98)*, volume 1547 of *Lecture Notes Comput. Sci.*, pages 72–86. Springer-Verlag, 1998.
25. R. Dorfman. The detection of defective members of large populations. *Ann. Math. Statist.*, 14:436–440, 1943.
26. D.-Z. Du and F. K. Hwang. *Combinatorial Group Testing and Its Applications*. World Scientific, 2nd edition, 2000.

27. U. Finkler and K. Mehlhorn. Checking priority queues. In *Proc. 10th ACM-SIAM Symp. on Discrete Algorithms*, pages S901–S902, 1999.
28. I. Gassko, P. S. Gemmell, and P. MacKenzie. Efficient and fresh certification. In *Int. Workshop on Practice and Theory in Public Key Cryptography (PKC '2000)*, volume 1751 of *LNCS*, pages 342–353. Springer-Verlag, 2000.
29. M. T. Goodrich. Efficient packet marking for large-scale IP traceback. In *9th ACM Conf. on Computer and Communications Security (CCS)*, pages 117–126, 2002.
30. M. T. Goodrich, M. Shin, R. Tamassia, and W. H. Winsborough. Authenticated dictionaries for fresh attribute credentials. In *Proc. Trust Management Conference*, volume 2692 of *LNCS*, pages 332–347. Springer, 2003.
31. M. T. Goodrich and R. Tamassia. Efficient authenticated dictionaries with skip lists and commutative hashing. Technical report, Johns Hopkins Information Security Institute, 2000. Available from `http://www.cs.brown.edu/cgc/stms/papers/hashskip.pdf`.
32. M. T. Goodrich, R. Tamassia, and J. Hasic. An efficient dynamic and distributed cryptographic accumulator. In *Proc. of Information Security Conference (ISC)*, volume 2433 of *LNCS*, pages 372–388. Springer-Verlag, 2002.
33. M. T. Goodrich, R. Tamassia, and A. Schwerin. Implementation of an authenticated dictionary with skip lists and commutative hashing. In *Proc. 2001 DARPA Information Survivability Conference and Exposition*, volume 2, pages 68–82, 2001.
34. M. T. Goodrich, R. Tamassia, N. Triandopoulos, and R. Cohen. Authenticated data structures for graph and geometric searching. In *Proc. RSA Conference—Cryptographers' Track*, pages 295–313. Springer, LNCS 2612, 2003.
35. D. Gross-Amblard. Query-preserving watermarking of relational databases and XML documents. In *ACM Symp. on Principles of Database Systems (PODS)*, pages 191–201, 2003.
36. V. Guruswami. *List Decoding of Error-correcting Codes*. PhD thesis, Massachusetts Institute of Technology, Boston, MA, 2001.
37. V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. In *IEEE Transactions on Information Theory*, pages 45:1757–1767, 1999.
38. B. Horne, L. Matheson, C. Sheehan, and R. Tarjan. Dynamic self-checking techniques for improved tamper resistance. In T. Sander, editor, *Security and Privacy in Digital Rights Management*, volume 2320 of *Lecture Notes in Computer Science*, pages 141–159. Springer-Verlag, 2002.
39. S. Khanna and F. Zane. Watermarking maps: Hiding information in structured data. In *ACM/SIAM Symp. on Discrete Algorithms*, pages 596–605, 2000.
40. V. King. A simpler minimum spanning tree verification algorithm. In *Workshop on Algorithms and Data Structures*, pages 440–448, 1995.
41. P. Kocher. A quick introduction to certificate revocation trees (CRTs), 1998. http://www.valicert.com/resources/whitepaper/bodyIntroRevocation.html.
42. P. C. Kocher. On certificate revocation and validation. In *Proc. Int. Conf. on Financial Cryptography*, volume 1465 of *LNCS*. Springer-Verlag, 1998.
43. I. Krsul and E. H. Spafford. Authorship analysis: Identifying the author of a program. *Computers and Society*, 16(3):248–259, 1997.
44. M. Kuhn. The trustno1 cryptoprocessor concept. Technical Report CERIAS-1997-04-30, Purdue University, 1997.
45. K. Kurosawa, T. Johansson, and D. R. Stinson. Almost k-wise independent sample spaces and their cryptologic applications. *Journal of Cryptology*, 14:231–253, 2001.
46. A. Lysyanskaya, R. Tamassia, and N. Triandopoulos. Multicast authentication in fully adversarial networks. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 241–255, May 2004.

47. C. Martel, G. Nuckolls, P. Devanbu, M. Gertz, A. Kwong, and S. G. Stubblebine. A general model for authenticated data structures. *Algorithmica*, 39(1):21–41, 2004.
48. K. Mehlhorn and S. Näher. *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, Cambridge, UK, 2000.
49. K. Mehlhorn, S. Näher, M. Seel, R. Seidel, T. Schilz, S. Schirra, and C. Uhrig. Checking geometric programs or verification of geometric structures. *Comput. Geom. Theory Appl.*, 12(1–2):85–103, 1999.
50. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.
51. J. Naor and M. Naor. Small-bias probability spaces: Efficient constructions and applications. In *ACM Symposium on Theory of Computing*, pages 213–223, 1990.
52. M. Naor and K. Nissim. Certificate revocation and certificate update. In *Proc. 7th USENIX Security Symposium*, pages 217–228, Berkeley, 1998.
53. R. Ostrovsky, C. Rackoff, and A. Smith. Efficient consistency proofs for generalized queries on a committed database. In *Proc. 31th International Colloquium on Automata, Languages and Programming (ICALP)*, 2004.
54. D. J. Polivy and R. Tamassia. Authenticating distributed data using Web services and XML signatures. In *Proc. ACM Workshop on XML Security*, 2002.
55. G. Qu and M. Potkonjak. Analysis of watermarking techniques for graph coloring problem. In *IEEE/ACM Int. Conf. on Computer-Aided Design*, pages 190–193, 1998.
56. D. Ron. Property testing. In P. M. Pardalos, S. Rajasekaran, J. Reif, and J. D. P. Rolim, editors, *Handbook of Randomized Computing*, pages 597–649. Kluwer Academic Publishers, 2001.
57. S. Savage, D. Wetherall, A. R. Karlin, and T. Anderson. Practical network support for IP traceback. In *SIGCOMM*, pages 295–306, 2000.
58. B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley and Sons, Inc., New York, NY, USA, second edition, 1996.
59. B. Schneier and J. Kelsey. Secure audit logs to support computer forensics. *ACM Trans. on Information and System Security*, 2(2):159–176, 1999.
60. R. Sion, M. J. Atallah, and S. K. Prabhakar. Resilient information hiding for abstract semi-structures. In *Proc. of the Workshop on Digital Watermarking (IWDW), Seoul, Korea*, LNCS. Springer-Verlag, 2003.
61. R. Sion, M. J. Atallah, and S. K. Prabhakar. Rights protection for relational data. In *Proc. 2003 ACM International Conference on Management of Data (SIGMOD), San Diego, California*, pages 98–109. ACM Press, 2003.
62. E. H. Spafford and G. Kim. The design and implementation of tripwire: A file system integrity checker. In *2d ACM Conf. on Computer and Communication Security (CCS)*, 1994.
63. E. H. Spafford and S. A. Weeber. Software forensics: Tracking code to its authors. *Computers and Society*, 12(6):585–595, 1993.
64. D. R. Stinson. *Cryptography: Theory and Practice, Second Edition*. CRC Press Series, 2002.
65. D. R. Stinson, T. van Trung, and R. Wei. Secure frameproof codes, key distribution patterns, group testing algorithms and related structures. *Journal of Statistical Planning and Inference*, 86:595–617, 2000.
66. G. F. Sullivan and G. M. Masson. Certification trails for data structures. In *Digest of the 21st Symposium on Fault-Tolerant Computing*, pages 240–247. IEEE Computer Society Press, 1991.
67. G. F. Sullivan, D. S. Wilson, and G. M. Masson. Certification of computational results. *IEEE Trans. Comput.*, 44(7):833–847, 1995.
68. R. Tamassia. Authenticated data structures. In *Proc. European Symp. on Algorithms*, volume 2832 of *Lecture Notes in Computer Science*, pages 2–5. Springer-Verlag, 2003.

69. R. Tamassia and N. Triandopoulos. Computational bounds on hierarchical data processing with applications to information security. In *Proc. Int. Colloquium on Automata, Languages and Programming (ICALP)*, LNCS. Springer-Verlag, 2005.
70. R. Venkatesan, V. Vazirani, and S. Sinha. A graph theoretic approach to software watermarking. In *4th Int. Workshop on Information Hiding*, volume 2137 of *LNCS*, pages 157–168. Springer-Verlag, 2001.
71. A. Yasinsac and Y. Manzano. Policies to enhance computer and network forensics. In *IEEE Workshop on Information Assurance and Security*, pages 289–295, 2001.