Nonadaptive Mastermind Algorithms for String and Vector Databases, with Case Studies

Arthur U. Asuncion and Michael T. Goodrich, *Fellow*, *IEEE*

Abstract—In this paper, we study sparsity-exploiting Mastermind algorithms for attacking the privacy of an entire database of character strings or vectors, such as DNA strings, movie ratings, or social network friendship data. Based on reductions to nonadaptive group testing, our methods are able to take advantage of minimal amounts of privacy leakage, such as contained in a single bit that indicates if two people in a medical database have any common genetic mutations, or if two people have any common friends in an online social network. We analyze our Mastermind attack algorithms using theoretical characterizations that provide sublinear bounds on the number of queries needed to clone the database, as well as experimental tests on genomic information, collaborative filtering data, and online social networks. By taking advantage of the generally sparse nature of these real-world databases and modulating a parameter that controls query sparsity, we demonstrate that relatively few nonadaptive queries are needed to recover a large majority of each database.

Index Terms—Mastermind algorithms, privacy leaks, data cloning, combinatorial group testing, nonadaptive attacks

1 Introduction

Privacy and data protection are important and growing concerns when dealing with character strings or vector data. Medical databases are constrained by Health Insurance Portability and Accountability Act (HIPAA) rules to keep identifying data private, for instance. Such databases in the future will commonly store DNA strings of patients, which will need to have their privacy protected for obvious reasons. Likewise, attribute vectors, which reflect the presence or absence of each of a large number of possible attributes, are common in biotechnology; for example, chemical attribute vectors (e.g., see [1], [2]) indicate the presence or absence of each of about a million attributes.

Privacy concerns also exist for online social networks and other databases which store user preferences in vector form. For instance, knowledge of a social network user's set of friends (representable as a row in an adjacency matrix) is potentially a gateway privacy leak, for friendship overlaps have been shown to be sufficient to deanonymize individuals across multiple social networking sites [3]. Likewise, the movie rating vectors in the database used for the Netflix Prize contest consist of ratings of movies by individual users, which are generally deemed as sensitive information. Full access to such databases may be constrained by privacy agreements or legitimate proprietary reasons for keeping these databases private, even as they allow for limited types of queries to be performed on them.

Each time a client queries such a database and it responds with an answer, it reveals some information about its contents, even if the client and the database are using a Secure Multiparty Computation (SMC) protocol (e.g., see [4], [5], [6], [7], [8], [9], [10], [11]) to process such a query. Thus, we can provide a crude characterization of the risk of privacy loss in biological, medical, or proprietary databases in terms of the existence of efficient algorithms that can take advantage of the data leakage present in query responses to be able to replicate part or all of the content of the database. We refer to such schemes as *data-cloning* attacks.

Formally, in an *algorithmic data-cloning attack*, a *querier*, Bob, is allowed certain types of queries to a database, $\mathcal{X}$, that belongs to a *data owner*, Alice. Bob's goal is to replicate all or a large part of $\mathcal{X}$ through as few queries on $\mathcal{X}$ as possible (and with low-computational overhead). In this paper, we focus on databases where $\mathcal{X}$ is a collection $\mathcal{X} = (X_1, X_2, \dots, X_g)$ of character strings or vectors, over a fixed-size alphabet. With respect to the types of databases we consider, we assume that Alice is willing to process *comparison* queries from Bob, each of which consists of Bob providing a single vector Q (which is not necessarily revealed in plaintext to Alice) and, possibly using a SMC. Alice reveals a response vector $(r_1, r_2, \dots, r_g)$, where each r_i is the score for some type of comparison of Q with X_i. In the simplest case, each score r_i can be a single bit denoting whether the query Q shares any common entries with X_i. As mentioned above, the risk to this *data-cloning* attack, then, can be characterized by the number of queries and how much processing time is needed so that Bob can replicate all of $\mathcal{X}$ or a large portion of $\mathcal{X}$.

1.1 Our Contributions

Inspired by a game known as *Mastermind*, we present a number of algorithms for performing a *Mastermind attack* on an entire string or vector database, $\mathcal{X} = (X_1, X_2, \dots, X_g)$, so as to clone all or a large portion of $\mathcal{X}$. All of our methods assume only the SMC protocol of Jiang et al. [8], where a querier, Bob, issues a query string or vector, Q, and receives a vector of responses $(r_1, r_2, \dots, r_g)$, where each r_i is a single

- *The authors are with the Department of Computer Science, Bren School of Information & Computer Sciences, University of California, Irvine, CA 92697-3435. E-mail: asuncion@uci.edu, goodrich@acm.org.*

Manuscript received 11 Dec. 2010; revised 26 May 2011; accepted 15 June 2011; published online 23 June 2011.
Recommended for acceptance by P. Ipeirotis.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2010-12-0659.
Digital Object Identifier no. 10.1109/TKDE.2011.147.

numerical response score measuring the similarity of Q and X_i according to some public metric. Since vectors taken over a universe of size c can be viewed as character strings taken over an alphabet of size c, we will, without loss of generality, focus our descriptions on the case when $\mathcal{X}$ consists of g character strings. We will also assume that each string in $\mathcal{X}$ is the same length, since we can view smaller strings as being padded with an additional character not in the original alphabet.

We show that repeated querying of such a database can clone all or a large portion of it, often with a surprisingly small, *sublinear* number of queries. The risk profile we explore in each type of attack, then, is the number of queries needed to execute it. Specifically, let us suppose that $\mathcal{X}$ contains g strings, each of length n, taken over an alphabet of size c, with at least $g' \leq g$ of these strings having at most $d < n$ differences from a public reference string, R (also of length n). We show that at least g' of the strings in $\mathcal{X}$ can be cloned using (at most) the following number of queries:

$$2(c-1)(2d \log n + \min\{d \log g, d^2 \log(en/d)\}).$$

This result applies to situations, common in many real-world databases (e.g., [1], [2], [12]), where strings in the database can be characterized in terms of a small number of differences with a reference string, R.

We also provide several case studies showing empirical data that demonstrates that our randomized attack can work effectively on real-world databases. For instance, we apply our attack to a database of mitochondrial DNA (mtDNA) strings and the database of movie-ratings vectors provided for the Netflix Prize contest, showing that large portions of these databases can be cloned using a number of queries that is much smaller than the length of the strings or vectors in these databases.

If, in practice, Bob learns more than the information contained in the response vector $(r_1, r_2, \ldots, r_g)$, that only strengthens his attack. The point of this paper is that even with just the information leaked in the responses, Bob can construct a small number of query vectors that are sufficient to learn all or a sizeable fraction of the vectors in $\mathcal{X}$. Moreover, our Mastermind attack is *oblivious* (that is, *nonadaptive*), in that Bob can construct all his query vectors in advance, so that the format of no query depends on the outcome of another. We describe a randomized construction for Bob's query vectors, which allows the attack to be fairly surreptitious, in that each query looks random (because it is random).

2 Attack Scenarios

Before describing our nonadaptive Mastermind attack in detail, we show how it applies to a wide variety of attack scenarios to provide motivating examples. We illustrate three such attack scenarios below.

2.1 Genetic Signatures

Suppose the vectors in $\mathcal{X}$ represent the genetic signatures of people in some population, such as a high school, college, or corporation. Bob's goal in this Mastermind attack is to learn the genetic signatures for as many people in his population of interest as is reasonably possible. He can employ his attack so long as there is a website or tool for $\mathcal{X}$ that allows him to test a query vector Q against the vectors in $\mathcal{X}$ to determine which ones share a mutation with Q, with respect to a reference R. In mitochondrial DNA, R is roughly 16,500 base pairs long, but has only about 4,000 known mutations [15], [16], suggesting that each vector in $\mathcal{X}$ is sparse relative to R.

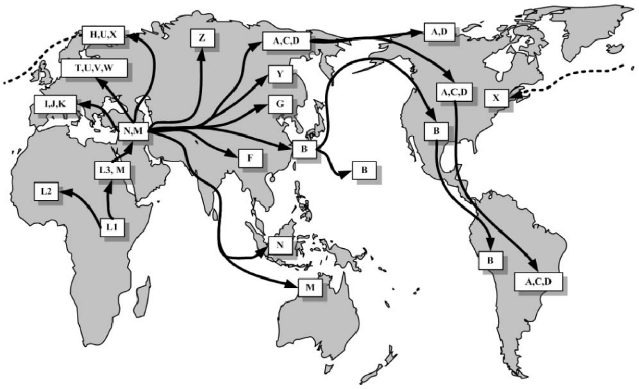


Fig. 1. An illustration of the pattern of human migration together with major mutations in human mtDNA [13], [14], which is only transferred along the maternal line. Each letter stands for a set of mutations from the reference string, R (which in this case is the Revised Cambridge Reference Sequence). Thus, determining locations of differences with R can reveal ethnic identity, sometimes to the resolution of the village of maternal ancestry. (Image, Copyright 2009, Michael T. Goodrich. Used with permission.)

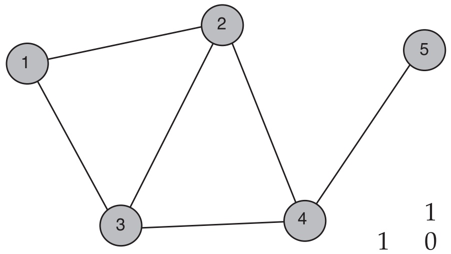
In this example, Bob could be posing as a medical researcher and claim that his vectors are testing for combinations of genetic markers for disease. Alternatively, he could claim to be a forensic analyst with DNA from a crime scene, which he wants to test against members of $\mathcal{X}$ (in this case, he is likely to receive a similarity score between his query Q and the vectors in $\mathcal{X}$, which he can easily convert into an overlap-detection bit). In either case, a minimum amount of overlap information can allow him to learn the entire genetic signatures of a large number of members of $\mathcal{X}$.

The privacy implications of such an innocuous attack are significant. Alice's genetic signature could then be used by an unethical employer or insurance company to discriminate against her based on her risks for future diseases. Also, as illustrated in Fig. 1, it is possible using a genetic signature derived from a short string of Alice's mitochondrial DNA to trace her maternal lineage to an ancestral location [13], [14], which is information that could then be used for ethnic discrimination [17].

2.2 Social Network Friendship Ties

Suppose the vectors in $\mathcal{X}$ represent the rows of the adjacency matrix (e.g., Fig. 2) defined by the friendship ties for an online social network, like Facebook, possibly restricted to the population in a specific city, college, high school, or large corporation. In this scenario, Bob wants to learn the friendship relationships of as many people as possible. For instance, he may wish to do racial profiling [18] or do a cross-networking identification attack [3], since 89 percent of Facebook users use their real names [19].

In this case, Bob's query vectors correspond to a relatively small number of pseudonyms that Bob creates



$$\begin{array}{cccccc} & 1 & 2 & 3 & 4 & 5 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 2 & 1 & 0 & 1 & 1 & 0 \\ 3 & 1 & 1 & 0 & 1 & 0 \\ 4 & 0 & 1 & 1 & 0 & 1 \\ 5 & 0 & 0 & 0 & 1 & 0 \end{array}$$

Fig. 2. An example graph and its adjacency matrix.

in the social network and for which he defines a certain number of random friendship ties. For instance, he could create such ties using automated social engineering techniques (e.g., using the name of an affiliated city, college, etc.) as well as the property that a fairly large percentage of social networking users are likely to accept random friendship requests from people in their community (roughly 10 to 25 percent of student Facebook users accept random friendship requests from people who say they are in the same university [20]). Given his set of pseudonyms, Bob employs the group testing attack by having each of his pseudonyms ask the social networking site if this pseudonym shares any friends with the people in Bob's population of interest. Note that he will receive a useful response vector from everyone that has privacy settings that allow for testing for friends in common. That is, even if someone chooses to share friendship information only with "friends of friends," which is one of the more restrictive standard privacy settings in Facebook, Bob can still get valid responses for his queries with respect to such people. Moreover, if Bob employs an oblivious group testing attack, he can use the same set of pseudonyms for everyone whose privacy he is attacking. Thus, once he has set up his pseudonyms, he can target the privacy of any user in the online social network at will.

2.3 User Preference Data

Suppose the vectors in $\mathcal{X}$ represent the preferences of people in a site, such as Amazon or Netflix, that employs collaborative filtering to support product recommendations. Specifically, we assume in this scenario that products are numbered 1 to k and each vector X_i in $\mathcal{X}$ has a discrete rating (e.g., 1-5 stars, or a missing rating) in position j, provided by user i. Bob's goal in this scenario is to discover as many vectors in $\mathcal{X}$ as reasonably possible and in so doing discover the product preferences of a large number of targeted people. His motivation could, for instance, be economic, in that he may want to open an online store that caters to a specific demographic; hence, Bob may want to learn the product preferences for a known population of people in this group. In terms of information leakage, all that is needed in order to allow for Bob's group testing attack to work is for the collaborative filtering site have a way for him to create pseudonyms, have these pseudonyms rate products, and allow for these pseudonyms to test if they have any ratings in common with users in the target population. So long as the collaborative filtering website allows for users to check for overlapping scores with other users, Bob can employ the nonadaptive Mastermind attack.

2.4 Exploiting Sparsity

The above set of attack scenarios are illustrative of the risks to privacy that the group testing attack provides, in that it can greatly amplify the information gained from a relatively small number of single-bit privacy leaks. The risk to the group testing attack can be characterized in terms of the number of queries and how much processing time is needed so that Bob can replicate a large portion of $\mathcal{X}$. As we will elaborate in Section 4, the critical factor is a sparsity parameter, d, which, in a group testing context, refers to the small number of "defective" items in the large group.

Interestingly, each of the attack scenarios mentioned above possesses such a parameter, allowing for Bob to employ efficient Mastermind attacks with a relatively small number of queries. For example, an individual's genetic signature will typically have a relatively small number of indicators for mutations with respect to a reference DNA string—with mitochondrial DNA, most people have fewer than 100 mutations with respect to a commonly used reference string. Furthermore, most people in social networking sites, such as Facebook, have less than a few hundred friends. Likewise, most collaborative filtering preference vectors, such as in the Netflix Prize contest, have ratings for at most a few hundred items. Thus, there are several modern contexts that have all the pieces in place to allow for the Mastermind attack to be used.

It is worth noting that realistic attacks can also be constructed in many other domains. For instance, sensitive image data, such as captured by biometric devices, may be represented as sparse vectors, making it susceptible to a Mastermind attack, especially when efficient tools exist for comparing a query (e.g., a fingerprint or an iris scan) to the entire database.

3 Background and Related Work

We give a brief background of the Mastermind game and attacks inspired by that game, as well as related work on privacy models and attempts to mitigate privacy leaks.

3.1 Mastermind

Adapting the terminology of the Mastermind attack [21] to attacks on an entire database, we discuss in this section the relationship between the Mastermind attack and the Mastermind game. Mastermind [22], [23] is a two-player board game, which is played between a *codemaker* and a *codebreaker*, using colored pegs (Fig. 3). Mastermind begins with the codemaker selecting a character string, X, of length n, using an alphabet of size c, whose members are called "colors." The codebreaker then makes a sequence of queries, $Q_1, Q_2, \ldots$, about X's identity. For each guess Q_i, the codemaker provides a score on how well Q_i matches X. In the board game, this is done using colored pegs, but we assume in this paper that the score is simply a matching function, $b(Q_i) = |\{j : Q_i[j] = X[j]\}|$, which counts the number of places where Q_i and X match. The codebreaker's goal is to discover X using a small number of guesses.

Chvátal [22] studied the combinatorics of the general Mastermind game, showing that it can be solved in polynomial time using $2n\lceil\log c\rceil + 4n$ guesses. Chen et al.

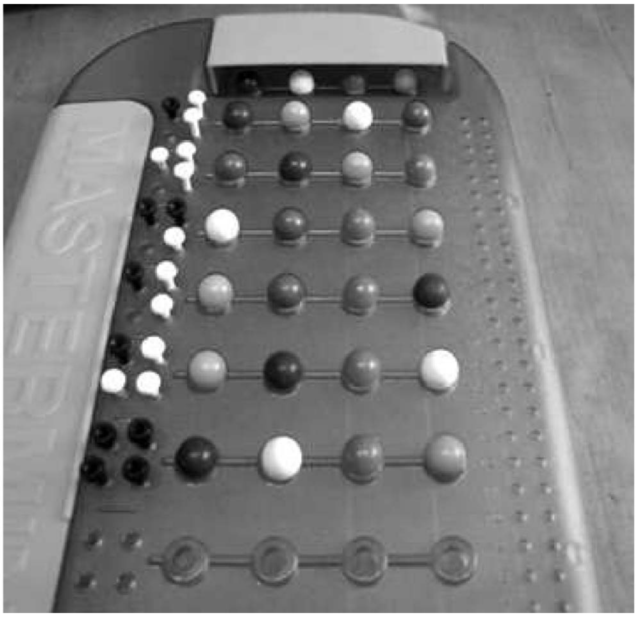


Fig. 3. The Mastermind game. The four large pegs in the middle are used for guessing. The four smaller peg locations on the left are used to score each guess, with black-peg and white-peg scores. (Image, Copyright 2009, Michael T. Goodrich. Used with permission.)

[24] showed how this can be improved to $2n\lceil\log n\rceil + 2n + \lceil c/n\rceil + 2$ guesses and Goodrich [25] showed how this bound can be improved to $n\lceil\log c\rceil + \lceil(2-1/c)n\rceil + c$. Unfortunately, from the perspective of the cloning problem, all of these algorithms are *adaptive*, in that they use results of previous queries to construct future queries. Adaptive algorithms can only be used effectively for the interactions between a single pair of strings. For a sequence of queries to be used against an entire database of strings, we need a *nonadaptive* algorithm, that is, an algorithm where queries are not dependent upon answers from previous queries, which is equivalent to the codebreaker making all his guesses in advance. Another benefit of this nonadaptive scheme is that all the queries can be performed in parallel.

Chvátal [22] also gives an existence proof for a nonadaptive method for solving Mastermind. If the number of possible colors, $c \le n^{1-\epsilon}$, for some constant $\epsilon > 0$, which will almost always be the case for biological databases, Chvátal shows the existence of a nonadaptive method using only

$$G = (2+\epsilon)n\frac{1+2\log c}{\log n - \log c},$$

guesses. In fact, he shows that making G guesses at random will be sufficient to determine a unique solution with high probability, using only the $b(Q_i)$ type of scores. Unfortunately, this existence proof does not immediately lead to a polynomial-time algorithm. Indeed, it is NP-complete to determine if a collection of Mastermind guesses with $b(Q_i)$ type of responses is satisfiable [25]. Nonetheless, in this paper, we will show that Mastermind attacks based on reductions to group testing can efficiently clone a sparse database using a sublinear number of nonadaptive queries.

3.2 Related Privacy Models

Following a framework by Bancilhon and Spyratos [26], Deutsch and Papakonstantinou [27] and Miklau and Suciu [28] give related models for characterizing privacy loss in information releases from a database, which they call *query-view security*. In this framework, there is a secret, $\mathcal{S}$, that the data owner, Alice, is trying to protect. Attackers are allowed to ask legal queries of the database, while Alice tries to protect the information that these queries leak about $\mathcal{S}$. While this framework is related to the data-cloning attack, these two are not identical, since in the data-cloning attack there is no specifically sensitive part of the data. Instead, Alice, is trying to limit releasing too much of her data to Bob rather than protecting any specific secret. Similarly, Kantarcioğlu et al. [29] study privacy models that quantify the degree to which data mining searches expose private information, but this related privacy model is also not directly applicable to the data-cloning attack.

There has been considerable recent work on data modification approaches that can help protect the privacy or intellectual property rights of a database by modifying its content. For example, several researchers (e.g., see [30], [31], [32], [33], [34], [35]) advocate the use of data watermarking to protect data rights. In using this technique, data values are altered to make it easier, after the fact, to track when someone has stolen information from a database. Of course, by that point, the data have already been cloned. Alternatively, several other researchers (e.g., [36], [37], [38], [39], [40], [41], [42], [43]) propose using *generalization* or *cell suppression* as methods for achieving quantifiable privacy-preservation in databases. These techniques alter data values to protect sensitive parts of the data, while still allowing for data mining activities to be performed on the database. We assume here that Alice is not interested in data modification techniques, however, for we believe that accuracy is critically important in several database applications. For example, even a single base-pair mutation in a DNA string can indicate the existence of an increased health risk.

As mentioned above, we allow for the queries Bob asks to be answered using SMC protocols, which reveal no additional information between the query string Q and each database string X_i other than the response score r_i. Such protocols have been developed for the kinds of comparisons that are done in genomic sequences (e.g., see [4], [44], [45]). In particular, Atallah et al. [4] and Atallah and Li [46] studied privacy-preserving protocols for edit-distance sequence comparisons, such as in the longest common subsequence (LCS) problem (e.g., [47], [48], [49]). Troncoso-Pastoriza et al. [50] described a privacy-preserving protocol for regular-expression searching in a DNA sequence. Jha et al. [7] give privacy-preserving protocols for computing edit distance and Smith-Waterman similarity scores between two genomic sequences, improving the privacy-preserving algorithm of Szajda et al. [9]. Aligned matching results between two strings can be done in a privacy-preserving manner, as well, using privacy-preserving set intersection protocols (e.g., see [10], [45], [51], [52], [53]) or SMC methods for dot products (e.g., see [6], [11], [54]). In addition, the Fairplay system [5] provides a general compiler for building such computations.

Du and Atallah [55] study an SMC protocol for querying a string Q in a database of strings, $\mathcal{X}$, as in our framework, where comparisons are based on approximate matching (but not sequence-alignment). Their SMC protocols for performing such queries provide a best match, not a score for each string in the database. Thus, their scheme would not be applicable in the attack framework we are considering in this paper. The SMC method of Jiang et al. [8], on the other hand, is directly applicable. It provides a vector of scores comparing a string (or vector) Q to a sequence of strings (or vectors), as we require in this paper. Thus, our Mastermind methods can be viewed as an attack on repeated use of the SMC protocol of Jiang et al.

Goodrich [21] studies the problem of discovering a single DNA string from a series of genomic Mastermind queries. All his methods are sequential and adaptive, however, so the only way they could be applied to the data-cloning attack on an entire biological database is if Bob were to focus on each string X_i in $\mathcal{X}$ in turn. That is, he would have to gear his queries to specifically discovering each X_i in n distinct "rounds" of computation, each of which uses a lot of string-comparison queries. Such an adaptation of Goodrich's Mastermind attacks to perform data cloning, therefore, would be prohibitively expensive for Bob. Our approach, instead, is based on performing a nonadaptive Mastermind attack on the entire database at once.

We note that others have investigated deanonymization techniques on both social networks [56] and Netflix data [57]. These works are complementary to our goal of cloning the databases themselves.

4 Exploiting Sparsity in an Algorithmic Data-Cloning Attack

In this section, we describe the details of our nonadaptive Mastermind data-cloning attack. It is often the case that all or a large fraction of the strings in a real-world string database can be characterized in terms of a small number of differences with a public reference string. In these situations, which are quite common, we can apply a reduction to nonadaptive group testing, which results in an efficient Mastermind attack as we will see.

4.1 Nonadaptive Combinatorial Group Testing

We briefly discuss the background behind nonadaptive combinatorial group testing and then develop the concepts of disinguishing and disjunct test matrices, before delving into the probabilistic construction of our attack algorithm.

4.1.1 Background

Group testing was introduced by Dorfman [58], during World War II, to test blood samples. The problem he addressed was to design an efficient way to detect the few thousand blood samples that were contaminated with syphilis out of the millions that were collected. His idea was to pool drops of blood from multiple samples and test each pool for the syphilis antigen. By carefully arranging the group tests and then discovering which groups tested positive and which ones tested negative he could then identify the contaminated samples using a small number of group tests (much smaller than the number needed to explicitly test each individual blood sample), thereby sparing thousands of G.I.'s from needless disease exposure. In this paper, we show that Dorfman's humanitarian discovery has an unfortunate dark side when it comes to privacy protection, for it enables privacy leaks to be amplified in a data-cloning attack.

In the *combinatorial group testing* problem (e.g., Du and Hwang [59]), one is given a set S of n items, at most d of which are "defective," for some parameter $d \leq n$, and one is interested in exactly determining which of the items in S are defective. One can form a test from any subset T of S and in a single step determine if T contains any defective items or not. If one can use information from the result of a test in formulating the tests to make in the future, then the method is said to be *adaptive*. If, on the other hand, one cannot use the results from one test to determine the makeup of any future test, then the method is said to be *nonadaptive*. For the application to the Mastermind attack, we are interested in nonadaptive methods.

There are several existing nonadaptive group testing methods [59], but these approaches are meant for a more general context than in our database cloning attack. In particular, these methods are designed to work for *any* set of items having d defective members. In our case, we are instead interested in specific sets of items that are derived from the database we are interested in cloning. Because of this, we can, in fact, derive improved bounds than would be implied by existing combinatorial group testing methods.

4.1.2 Distinguishing and Disjunct Test Matrices

Suppose we are given a collection, $\mathcal{C}$, of sets, $\mathcal{C} = \{S_1, S_2, \ldots, S_g\}$, which are not necessarily distinct, such that each set S_i contains n items, at most d of which are "defective." We want to design a nonadaptive group testing scheme that can exactly identify the subset, D_i, of at most d defective items in each set S_i in $\mathcal{C}$. Our approach to solving this problem is based on a randomized approach used by Eppstein et al. [60].

A nonadaptive group testing algorithm can actually be viewed as a $K \times n$ $0-1$ matrix, M. Each of the n columns of M corresponds to one of the n items and each of the K rows of M represents a test. If $M[i,j] = 1$, then item j is included in test i, and if $M[i,j] = 0$, then item j is not included in test i. Since this is a nonadaptive testing scheme, we assume that no test depends on the results of any other. That is, every row of the matrix M is defined in advance of any test outcomes. The analysis question, then, is to determine how many tests are needed to provide useful results.

Let C denote the set of columns of M. Given a subset D of d columns in M, and a specific column j in C but not in D, we say that j is *distinguishable* from D if there is a row i of M such that $M[i,j] = 1$ but i contains a 0 in each of the columns in D. If each column of M that is in C and not in D is distinguishable from D, then we say that M is *D-distinguishing*. Furthermore, we generalize this definition, so that if M is D_i-distinguishing for each subset, D_i, in a collection, $\mathcal{D} = \{D_1, D_2, \ldots, D_g\}$, of columns in C, then we say that M is $\mathcal{D}$-distinguished. Finally, we say that the matrix M is *d-disjunct* (e.g., see Du and Hwang [59, p. 165],) if it is $\mathcal{D}$-distinguished for the collection, $\mathcal{D}$, of all of the $\binom{n}{d}$ subsets of size d of C.

Note that if M is D-distinguishing, then it leads to a simple testing algorithm with respect to D. In particular,

TABLE 1
Description of Commonly Used Variables

$\mathcal{X}$	Database of strings
g	Number of strings in the database
n	Length of each string
c	Size of alphabet (i.e., number of "colors")
R	Reference string
d	Maximum number of differences from R

TABLE 2
An Example of a $\mathcal{D}$-Distinguished Matrix M for the Network in Fig. 2

1	0	0	0	0
0	0	0	1	0
0	0	0	0	1
0	1	0	0	0
0	1	0	0	1
0	0	0	0	1
0	0	1	0	0
1	0	0	1	0
0	0	0	0	1
1	0	1	0	0
0	1	0	0	1

Here, $\mathcal{D} = \{D_1, D_2, \dots, D_5\}$, where D_i corresponds to the subset of "defective" columns (i.e., those that contain a "1") in row i of the adjacency matrix. We split the matrix M below into five regions to show the tests that distinguish each D_i.

suppose D is the set of defective items and we perform all the tests in M. Note that, since M is D-distinguishing, if an item j is not in D, then there is a test in M that will determine the item j is not defective, for j would belong to a test that must necessarily have no defective items. So we can identify D in this case—the set D consists of all items that have no test determining them to be nondefective.

Of course, if M is d-disjunct, then this simple detection algorithm works for any set D of up to d defective items in C. Unfortunately, building such a matrix M that is d-disjunct requires M to have $\Omega(d^2 \log n / \log d)$ rows [59], [61]. So we will instead build a matrix that is $\mathcal{D}$-distinguished for the collection, $\mathcal{D}$, of defective subsets determined by the sets of items in $\mathcal{C}$, with high probability.

Before delving into the probabilistic construction of M, we show in Table 2 a simple example of a $\mathcal{D}$-distinguished matrix M for the social network example in Fig. 2. Here, $\mathcal{D} = \{D_1, D_2, \dots, D_5\}$, where each D_i corresponds to row i of the adjacency matrix in Fig. 2; namely, D_i is the subset of columns in row i that are defective (i.e., have a "1"). We visually split the matrix M into five regions to show the tests that distinguish each D_i. It is important to note that the matrix M in Table 2 is simply shown for illustrative purposes; indeed, a simpler example of a $\mathcal{D}$-distinguished M is the identity matrix, which in this case would only require $n = 5$ tests. We discuss this simple baseline more in the experimental section. In general, the attack would need a number of tests that is sublinear in the string length n, as we will show in the following sections.

4.1.3 A Probabilistic Construction of the Test Matrix

Given a parameter t, which is a multiple of d, we construct a $2t \times n$ matrix M as follows: For each column j of M, we choose t/d rows uniformly at random and we set the values of these entries to 1, with the other entries in column j being set to 0. Note, then, that for any set D of up to d defective items, there are at most t tests that will have positive outcomes (detecting defectives) and, therefore, at least t tests that will have negative outcomes. Our desire, of course, is that columns that correspond to samples that are distinguishable from the defective ones should belong to at least one negative-outcome test. So, let us focus on bounds for t that allow for such a matrix M to be chosen with high probability.

Let C be a set of (column) items having a fixed subset D of d defective items. For each (column) item j in C but not in D, let Y_j denote the 0-1 random variable that is 1 if j is falsely identified as a defective item by M (that is, j is not included in a test of items distinguished from those in D). Let Y_j be 0 otherwise. Observe that the Y_j's are independent, since Y_j depends only on whether the choice of rows we picked for column j collide with at most t rows of M picked for the columns corresponding to items in D. There are a total of $2t$ rows, at most t of which contain a test with a defective item. Thus, the probability of any nondefective item joining any particular test having a defective item in it is at most $1/2$; hence, any Y_j is 1 (a false positive) with probability at most $2^{-t/d}$, since each item is included in t/d tests at random.

Let $Y = \sum_{j=1}^{n} Y_j$, and note that the expected value of Y, $E(Y)$, is at most $\hat{\mu} = n/2^{t/d}$. Thus, if $\hat{\mu} \leq 1$, we can use Markov's inequality to bound the probability of the (bad) case when Y is nonzero as follows:

$$\Pr(Y \geq 1) \leq E(Y) \leq \hat{\mu} = \frac{n}{2^{t/d}}.$$

Thus, if we set

$$t \geq 2d \log n,$$

then M will be D-distinguishing with probability at least $1 - 1/n$, for any particular subset of defective items, D, from a set C of n items. Likewise, the probability of having at least one false positive across g subsets is bounded by $ng/2^{t/d}$; thus, if we set

$$t \geq 2d \log n + d \log g,$$

then M will be $\mathcal{D}$-distinguished, with probability at least $1 - 1/n$, for the collection of g subsets of defective items determined by the sets in $\mathcal{C}$. Finally, we can use the fact (e.g., see Knuth [62]) that

$$\binom{n}{d} < (en/d)^d,$$

so that if we set

$$t \geq 2d \log n + d^2 \log(en/d),$$

then M will be d-disjunct with probability at least $1 - 1/n$, which implies M will work for any subset of at most d defective items. Therefore, we have the following.

Theorem 1. *If*

$$t \geq 2d \log n + \min\{d \log g, d^2 \log(en/d)\},$$

then a $2t \times n$ random matrix M, constructed as described above, is $\mathcal{D}$-distinguished, with probability at least $1 - 1/n$, *for any given collection, $\mathcal{D} = \{D_1, D_2, \ldots, D_g\}$, of g subsets of size d of the n columns in M.*

Proof. Let $\mathcal{D}$ be a given collection of g (not necessarily distinct) subsets of size d of the n columns in M. If

$$d^2 \log(en/d) > d \log g,$$

then M is $\mathcal{D}$-distinguished by construction, with probability at least $1 - 1/n$. If, on the other hand,

$$d^2 \log(en/d) \leq d \log g,$$

then M constructed as above is d-disjunct, with probability at least $1 - 1/n$, which implies it is $\mathcal{D}$-distinguished w.h.p. for any collection $\mathcal{D}$ of subsets of size d of the n columns of M. □

As mentioned above, this is a way of constructing a simple nonadaptive group testing method for identifying the defective items in the collection, $\mathcal{D}$, of subsets of up to d defective items determined by the sets in $\mathcal{C}$.

4.2 Reducing Mastermind to Group Testing

In this section, we describe how to use nonadaptive group testing to construct an efficient Mastermind cloning attack. Consider the case when $\mathcal{X}$ is a database of g strings of length n each, with each of them having at most $d \leq n$ differences with a reference string R. We assume that each string in $\mathcal{X}$ is drawn from an alphabet of c characters (or "colors"), which, without loss of generality, we assume are integers in the range $[0, c-1]$.

Suppose, like before, that we have a $2t \times n$ nonadaptive group testing matrix, M, for a set of size n having at most d defectives, where

$$t \geq 2d \log n + \min\{d \log g, d^2 \log(en/d)\}.$$

As before, we begin our general Mastermind cloning attack by making a query for the reference string, R. Let r be the response score for the query for R. Next, we create $c-1$ different string queries, $Q_{k,l}$, for each of the $K = 2t$ tests in M (where k denotes the row index in M), defined, for $l = 1, 2, \ldots, c-1$, as follows:

$$Q_{k,l}[j] = \begin{cases} R[j] & \text{if } M[k,j] = 0 \\ (R[j] + l) \bmod c & \text{else.} \end{cases}$$

Each such query against a string X_i will have some response, $r_{k,l,i}$. We interpret test (k, l, i) as having a "positive" response, that is, it does not detect a defective, if, in making the comparison of $Q_{k,l}$ with the string X_i, the response

$$r_{k,l,i} = r - b_{k,0,i},$$

where $b_{k,0,i}$ is the number of characters in X_i matching their associated (color-0) location in R at places where there are 1's in row k of M. Intuitively, each 1 in row k of M indicates a place where we test a deviation from the reference value in R at that location to the color l away (i.e., a color equal to $(R[j] + l) \bmod c$). If none of these locations is a match with the current X_i string, then none of these locations take a color that is l additive colors from their reference value. In other words, defective "items" in the associated group testing method correspond to locations where X_i differs from the reference string with characters that are exactly l away from their reference values.

Of course, being able to determine if such a test for $Q_{k,l}$ against string X_i is "positive" or "negative" requires that we know the value $b_{k,0,i}$, which we don't immediately know. We do immediately know the number, b_k, of 1's in row k of M, however. And, after we perform the queries for each $Q_{k,l}$ against a string X_i, we learn each response $r_{k,l,i}$. That is, we have c linear equations in c unknowns from these queries and their responses. Specifically, we have the equation, $b_k = b_{k,0,i} + b_{k,1,i} + \cdots + b_{k,c-1,i}$, where $b_{k,l,i}$ denotes the number of places j where there is a 1 in row k of M and the character in position j of X_i is l away from the reference, that is, places where $X[j] = (R[j] + l) \bmod c$ and $M[k,j] = 1$. We also have $c-1$ equations,

$$r_{k,l,i} = r - b_{k,0,i} + b_{k,l,i},$$

for $l = 1, 2, \ldots, c-1$, which can each be rewritten as $b_{k,l,i} = r_{k,l,i} - r + b_{k,0,i}$. This allows us to rewrite

$$b_k = c\, b_{k,0,i} - (c-1)r + \sum_{l=1}^{c-1} r_{k,l,i}.$$

Thus, we can determine the value of $b_{k,0,i}$ as

$$b_{k,0,i} = \frac{b_k + (c-1)r - \sum_{l=1}^{c-1} r_{k,l,i}}{c},$$

which in turn tells us which of the $Q_{k,l}$ tests are "positive" and which ones are "negative." Essentially, we are performing a combinatorial group test for each possible shift we can make from a color in reference R.

Thus, if there are at most d locations where X_i differs from the reference string and M is $\mathcal{D}$-distinguished for the set of at most d locations of difference for each string in $\mathcal{X}$, then this scheme will learn the complete identity of each string in $\mathcal{X}$. That is, this method will clone $\mathcal{X}$, with high probability. Therefore, by Theorem 1, we have the following:

Theorem 2. *Given a database $\mathcal{X} = (X_1, X_2, \ldots, X_g)$, of strings of length n defined over an alphabet of size c, there is a nonadaptive Mastermind cloning method that can discover each string in $\mathcal{X}$, using $2t(c-1)$ tests, with probability at least $1 - (c-1)/n$, where t is the smallest multiple of d such that*

$$t \geq 2d \log n + \min\{d \log g, d^2 \log(en/d)\},$$

and $d \leq n$ is the maximum number of differences any string in $\mathcal{X}$ has with R.

5 Case Studies

To test the real-world risks of the nonadaptive Mastermind cloning attack, we applied our methods to case studies involving random samples from a number of real-world string and vector databases, including genomic and social network data. We briefly describe the data sets used and then discuss experimental results which reveal that relatively few tests are needed to recover large proportions of each database.

TABLE 3
Characteristics of Data Sets Used in This Paper

Name	Strings	Length	Max Diff	Colors
Genomic	457	16,568	492	4
Netflix	1,000	17,700	1988	6
Epinions	2,000	131,827	517	3
Slashdot	2,000	82,144	378	3
Slashdot (All)	82,144	82,144	428	3
Facebook-UNC	18,163	18,163	3,795	2
Facebook-Unif	1,000	72,261,577	2,164	2

5.1 Data Sets

We analyze several different data sets with varying characteristics to test our approach. For each data set, Table 3 shows the number of strings g, string length n, maximum difference d from the reference R across strings (where "difference" is defined as the number of entries in which the string differs from R), and the number of unique colors c present in the database.

The Genomic database consists of 457 human mitochondrial sequences downloadable from GenBank.[1] We use the Revised Cambridge Reference Sequence (rCRS), of length 16,586 bp as the reference string R. Fig. 4 shows the distribution of sequence differences from R, which reveals that the differences from R are relatively few and are concentrated at several different modes. In this data, there are four colors, namely the nucleotides A, C, T, and G.

Our movie-rating database is taken from the Netflix Prize data,[2] which consists of 100 million movie ratings and 480,189 different Netflix users. In our experiments, we use a representative subset of 1,000 randomly selected users. Each user has an associated string over 17,770 movies, where each position i stores the rating (from 1 to 5) given by the user for movie i. An entry of 0 signifies that the user has not rated that movie. Thus, there are six different unique colors in this database (0-5). Our reference R consists of all zeros, representing the case where no movies are rated. Fig. 4 shows that most users rate less than 300 movies. This sparsity allows the Mastermind attack to be very efficient, as we will see in the experiments.

We also analyze online social networks such as Epinions, Slashdot, and Facebook. Available from the SNAP Library,[3] Epinions, and Slashdot are "signed" networks, where positive and negative links appear in the network's adjacency matrix [63]. The Epinions network is the site's "Web of Trust" where users specify the other users that they trust or distrust. Similarly, in the Slashdot network, users can specify both "friends" and "foes." Hence, in both these databases, there are three unique colors: 0 (no link), 1 (good link), and -1 (bad link). In our experiments for both Epinions and Slashdot, we select a random subset of 2,000 users and utilize the corresponding rows in the adjacency matrix as our database. We also simulate a single large-scale group testing attack on the entire Slashdot-All adjacency matrix with 82,144 users.

The two Facebook data sets that we analyze are Facebook-Uniform and Facebook-UNC. Facebook-Uniform, provided by the authors of [64], is an unbiased sample of 957K unique users obtained by performing Metropolis-Hastings random walks over the Facebook network. Each user is associated with a (sparse) binary vector of size 72 million which denotes adjacencies. We restrict ourselves to a random subset of 1,000 users in Facebook-Uniform. Meanwhile, Facebook-UNC is a self-contained Facebook network of approximately 18,000 students at the University of North Carolina at Chapel Hill [65].

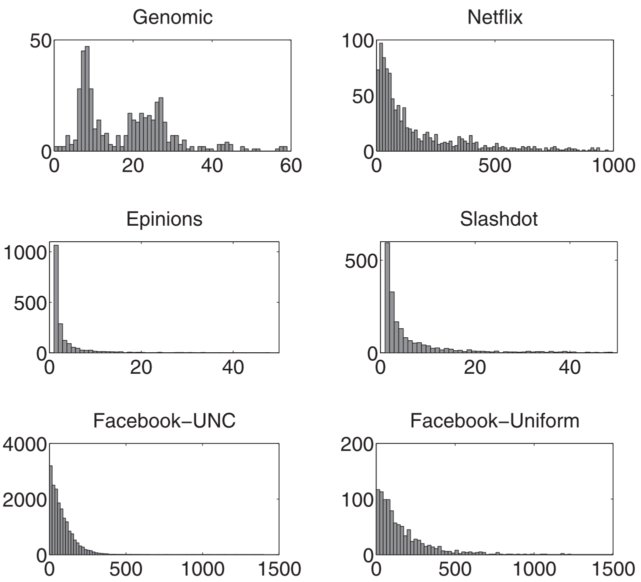


Fig. 4. Histogram of differences from reference R, for each of the data sets.

For all the social network data sets, we use a reference string R of all zeros. Fig. 4 shows that these networks are also sparse, which is often the case in many real-world settings.

5.2 Experiments

We briefly detail the experimental setup and discuss a baseline method as well as the theoretical number of Mastermind tests needed to clone each data set. Then, we investigate the effect of modulating a sparsity parameter and show that we can clone a large majority of each database with a relatively small number of tests.

5.2.1 Experimental Setup and Baselines

Our experimental approach is based on the analysis in Section 4. Similar to randomly selecting $\frac{t}{d}$ rows from $2t$ rows (for each column in the nonadaptive group matrix M), we stochastically set each entry in M to 1 with probability $p = \frac{1}{2d}$. This procedure enables us to add additional tests to M until the string is cloned or until an arbitrary cutoff of $100{,}000 * c$ tests is reached, where c is the number of unique colors in the database. We initialize with the same random seed for each string, ensuring that the same exact tests are performed on each string. This scheme allows us to determine the actual number of tests needed to clone the strings.

Before delving into the experimental results, we report in Table 4 the theoretical number of tests needed to clone the entire database with high probability, using the nonadaptive Mastermind technique. These numbers are based on n, g, d, c, and the bound in Theorem 2. Table 4 also shows the

[1] http://www.ncbi.nlm.nih.gov/Genbank/index.html.
[2] http://www.netflixprize.com.
[3] http://snap.stanford.edu/data/index.html#signnets.

TABLE 4
Theoretical Number of Tests Needed to Clone Entire Database (a) by Baseline Method (b) by Nonadaptive Mastermind Attack

	Baseline	Mastermind
Genomic	49,704	76,752
Netflix	88,500	536,760
Epinions	263,654	66,176
Slashdot	164,288	46,872
Slashdot (All)	164,288	58,208
Facebook-UNC	18,163	227,700
Facebook-Uniform	72,261,577	190,432

number of tests needed by a baseline technique to exactly clone the entire database. This baseline technique generates tests based on the reference R. For each entry j within R, and for each color offset l, a test is created where the entry j in R is replaced with its color offset l, namely $(R[j] + l)\ mod\ c$. Thus, the baseline method needs $(c - 1) * n$ tests to recover the entire database. Interestingly, the baseline technique can beat the theoretical bound (with d) when n is small, as is the case for the Genomic, Netflix, and Facebook-UNC data. However, the next section shows that modulating a sparsity parameter enables the attack to outperform both the baseline and the theoretical bounds.

5.2.2 The Effect of the Sparsity Parameter

Fortunately, our Mastermind attack can take advantage of the sparsity in the data to improve its efficiency. Since each string's distance from R is usually much smaller than d, it is empirically advantageous to use a target $\hat{d}$ that is much smaller than d. For instance, the Netflix data have a maximum difference $d = 1{,}988$, but the mean difference from R is $d_{\text{mean}} = 202$ and the median is $d_{\text{median}} = 92$. Thus, there are different possible choices for $\hat{d}$.

For each data set (excluding Slashdot-All and Facebook-Uniform due to their large scale), Fig. 5 shows the number of tests needed to exactly clone a string (averaged across all strings in the database), as a function of $\hat{d}$. In a few instances, when the strings are very far from R, the algorithm may reach the cutoff value, causing the mean to be undervalued; thus, we also plot the median number of tests since the median is more robust to outliers. Generally, we see that mean and median number of required tests decreases as $\hat{d}$ is decreased from d. For instance, for the Slashdot database, the mean/median number of tests is 18,000 if $\hat{d} = d = 378$, but if $\hat{d} = 50$, the mean/median number of tests is 3,000 and if $\hat{d} = d_{\text{mean}} = 13$, the median number of tests required is 700. Sometimes, the mean number of tests increases if $\hat{d}$ is too small though. If $\hat{d} = d_{\text{mean}} = 13$, the mean number of tests required is around 4,000. Thus, there is a tradeoff. If $\hat{d}$ is too small, it would take longer to exactly clone a string that is far away from R. If $\hat{d}$ is too large (e.g., $\hat{d} = d$), then many inefficient tests would be performed on strings that are close to R. We assume that a good estimate for $\hat{d}$ (such as the median distance from R) can be obtained a priori, e.g., through scientific knowledge in the case of the Genomic database, or publicly available information in the cases of Netflix, Epinions, Slashdot, and Facebook.

We also investigate the relationship between the number of required tests and the vector's distance from R. In Fig. 6, we observe that the number of tests required to clone a vector is very low (and nearly constant) when the vector's distance from R is itself low and close to $\hat{d}$. As the vector's distance increases, the number of required tests grows more quickly due to the mismatch between the distance and $\hat{d}$. For each data set, we display different scatter plots for different settings of $\hat{d}$. For instance, for the Slashdot data, the number of tests is relatively constant across all distances when the $\hat{d} = 200$; however, at this setting, the number of required tests is at least 10,000, even when the vector is close to the reference R. In contrast, when $\hat{d} = 3$, the number of required tests is only in the hundreds, around the vicinity

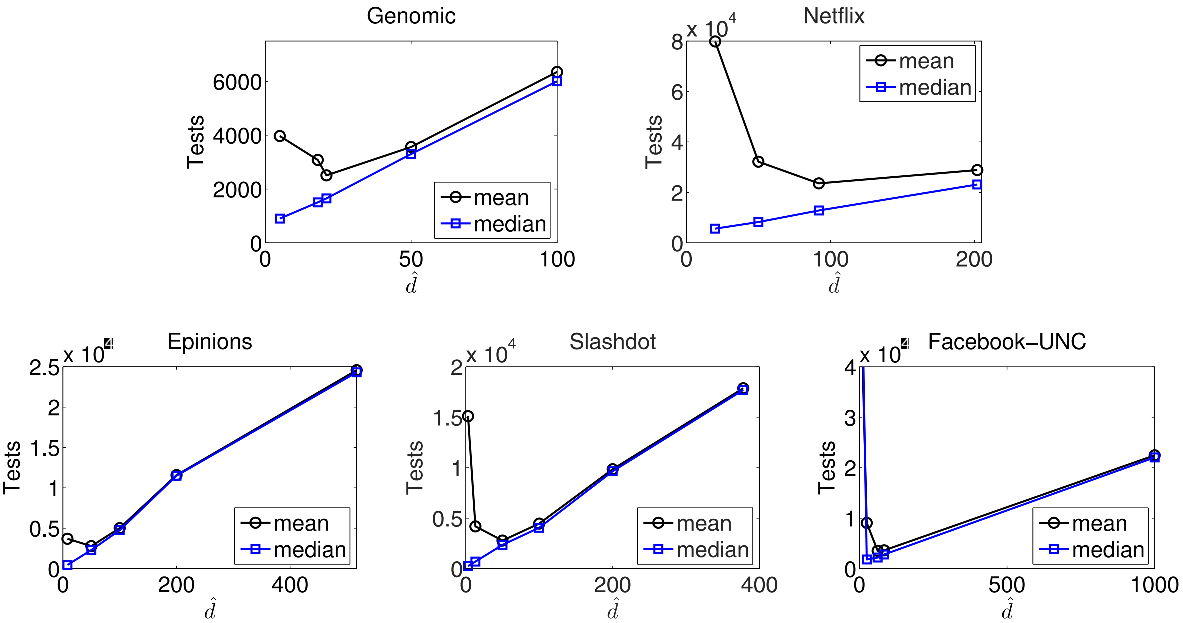


Fig. 5. Mean and median number of tests required until string is cloned (averaged across all strings in database), for various settings of target distance $\hat{d}$. Typically, it is advantageous to set $\hat{d}$ to be much less than d, since most of the vectors are sparse and are close to the reference R.

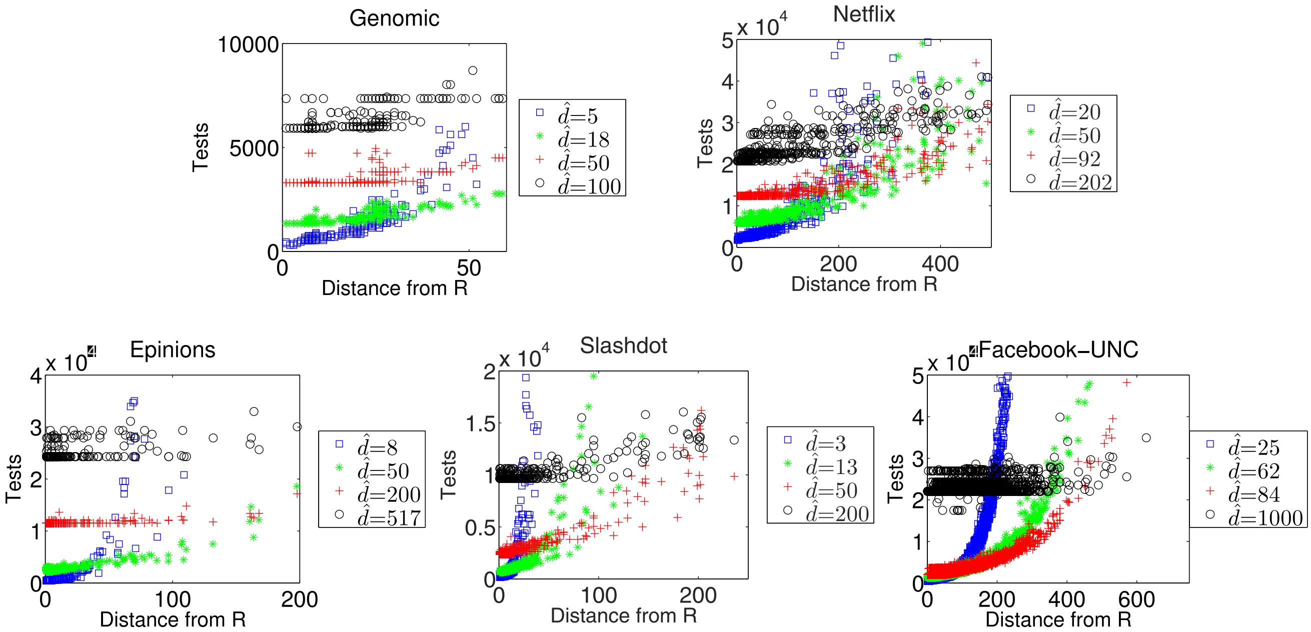


Fig. 6. Number of tests required to clone each string, ordered by the string's distance from R. Each string is represented by a dot. While the number of tests increases rapidly for small $\hat{d}$ when the vector is far from R, note that many vectors are close to R, allowing for a majority of the database to be captured quickly.

of $\hat{d}$; however, when the vector's distance from R is significantly greater (e.g., over 100), the scatter plot increases dramatically. It is important to note that most vectors are close to R due to the sparsity of the data, and thus, even when the scatter plot dramatically increases when the distance from R is great, there are relatively few vectors that fall within this regime.

Providing another perspective, Fig. 7 shows the decrease in error (defined as the number of differences between the string and the state of the reconstructed string) as the number of tests increases, for a randomly selected Netflix user who has rated 68 movies. One can see that using $\hat{d} = 202$ induces a slower rate of convergence than when using smaller settings for $\hat{d}$. The case where $\hat{d} = d = 1{,}988$ is not shown since its rate of convergence is even slower.

5.2.3 Cloning Results

In Fig. 8, the percentage of the database cloned by the nonadaptive Mastermind attack is plotted as a function of the number of tests, for various $\hat{d}$. We highlight some examples which demonstrate the efficiency of this attack. For the Genomic data (using $\hat{d} = d_{\text{median}} = 18$), 78 percent of the database is successfully recovered after 2,000 tests, and over 99 percent of the database is recovered after 3,000 tests, which is significantly less than both the baseline result (49,704 tests) and the theoretical bound (76,752 tests) in Table 4. For the Netflix data (using $\hat{d} = d_{\text{median}} = 92$), 63 percent of the strings are recovered after 10,000 tests. For the Epinions data (using $\hat{d} = d_{\text{mean}} = 8$), 68 percent of the strings are recovered after only 500 tests. For the Slashdot data (using $\hat{d} = d_{\text{mean}} = 13$), 82 percent of the strings are recovered after only 1,000 tests, which is much less than the number of tests required by the baseline method (164,288 tests) or the theory (46,872 tests) in Table 4.

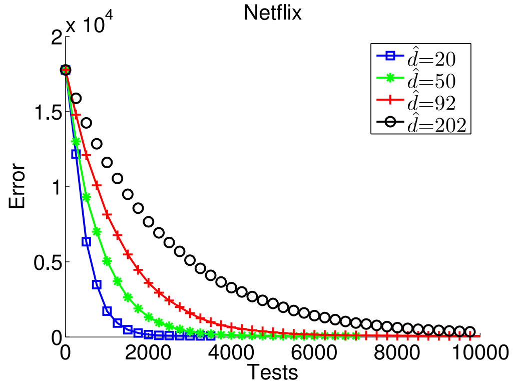


Fig. 7. Error as a function of the number of tests for a single Netflix user who has rated 68 movies, for various $\hat{d}$.

For Facebook-UNC, we see that the Mastermind attack displays different behavior for different choices of $\hat{d}$. When $\hat{d} = 5$, the attack is able to quickly recover (the sparsest) 15 percent of the data set after only 500 tests, but as the number of tests increases, the rate of progress slows significantly. When $\hat{d} = 25$, 52 percent of the database has been successfully recovered after 2,000 tests. Thus, using only a couple thousand nonadaptive tests, we are able to clone the friend lists of half (9K out of 18K) of the Facebook users at the University of North Carolina.

We also performed a large-scale nonadaptive Mastermind attack on Slashdot-All with 82,144 users. Fig. 9 shows that 55 percent of the strings are recovered after 2,500 tests and that 81 percent of the strings are recovered after 4,000 tests, using $\hat{d} = 50$. In contrast, the theoretical Mastermind bound suggests that 58,208 tests are needed to clone the entire Slashdot-All database, while the baseline method would need 164,288 tests, as indicated in Table 4. Even when using a $\hat{d}$ which may be suboptimal, our empirical results suggest that it is possible to substantially outperform both the baseline method as well the theoretical

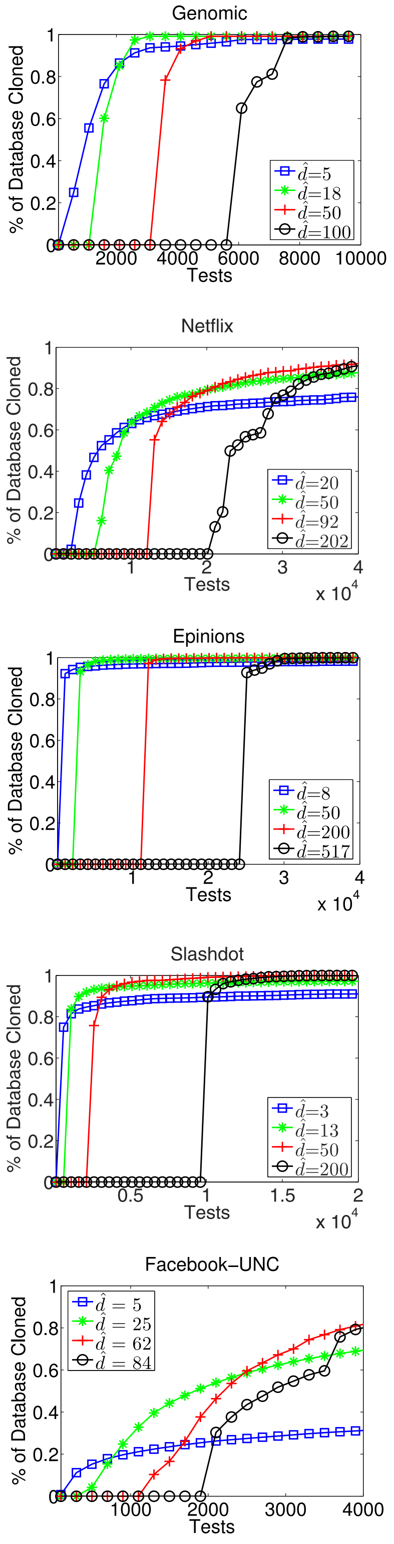


Fig. 8. Percentage of strings cloned as a function of the number of tests, for each data set, using various $\hat{d}$.

bounds in Table 4 in practice, as long as $\hat{d}$ is chosen to be less than d.

We also ran the same experiment on Facebook-Uniform for $\hat{d} = 108$ (the median distance from R). Fig. 9 shows that over 70 percent of the data set can be reconstructed with 10,000 tests, despite the fact that the vector length of this data set is huge ($n = 2{,}261{,}577$). Since Facebook-Uniform contains an unbiased sample of users, these users are representative of the global Facebook population. Furthermore, our theory states that the number of required tests increases at a rate of at most $\log(g)$ where g is the number of Facebook users. In fact, the theoretical number of tests needed to guarantee that 50 percent of a 300-million user Facebook network is cloned is less than 20,000 (assuming $d_{\text{median}} = 130$).[4] These results imply that an attacker may be able to recover over half of the global Facebook social network with several thousands of seemingly innocuous nonadaptive Mastermind queries.

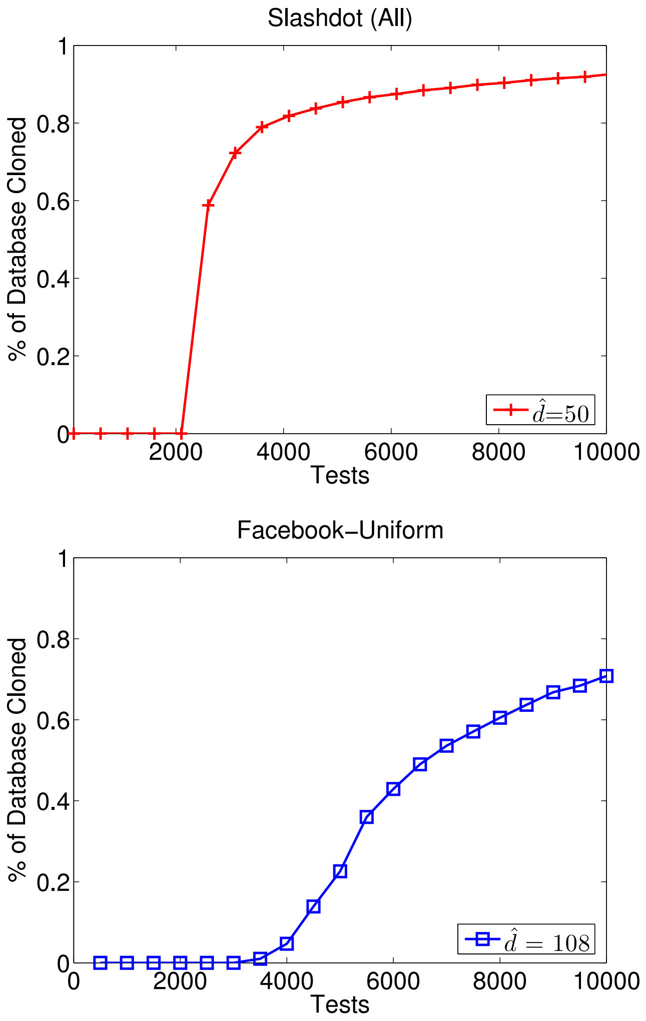


Fig. 9. Percentage of strings cloned as a function of the number of tests, for the large-scale data sets. Slashdot-All has a large number of strings ($g = 82{,}144$) while Facebook-Uniform has large vector length ($n = 72{,}261{,}577$).

It is worth noting that experiments have also been conducted on a variety of other data sets not mentioned in this paper—the nonadaptive Mastermind attack also performs very well on those data sets. Results on cloning databases of binary attribute vectors (i.e., where the number of colors $c = 2$) are described in previous work [66].

Our empirical results have shown that there is sensitivity to the choice of $\hat{d}$ in certain cases. One possible improvement is to use a tiered approach, where $\hat{d}_1$ is used to

[4] According to http://www.facebook.com/press/info.php?statistics, $d_{\text{mean}} = 130$, and so d_{median} should be even smaller, suggesting that the Mastermind attack can be even more efficient.

construct the first 5,000 tests, $\hat{d}_2$ is used to construct the next 5,000 tests, etc. This tiered approach may also be useful for databases with multimodal data, in which some data cases are close to the reference string, while other cases are far away from the reference string; in this scenario, each $\hat{d}_i$ could correspond to a different mode. Nonetheless, even when using a single $\hat{d}$, our results demonstrate that it is possible to clone a large fraction of a sparse database, by simply performing a nonadaptive Mastermind attack.

6 Conclusion

We have studied the Mastermind cloning attack, both from a theoretical and experimental perspective, and have shown its effectiveness in being able to copy the contents of a string database through a sublinear number of string-comparison queries. Furthermore, our approach benefits from being fully nonadaptive and surreptitious in nature (due to the randomized query construction), which is useful in real-world settings.

An interesting future research direction would be to allow for minimal adaptivity among the queries. For instance, in the case of social networks, one may allow each fake user to slightly modify its settings based on the previous set of similarity results across all the vectors in the database. In this scenario, it is likely that fewer tests will be needed; however, we also lose the benefit of being able to perform all the tests in parallel (or in any order).

Another natural direction for future work, of course, is on methods for defeating our nonadaptive Mastermind attack, which we have not addressed in this paper. Certainly, having Alice randomly permute the responses from her database with each query could help, since it would make it harder (but not necessarily impossible) for Bob to correlate responses between different queries. Of course, requiring Alice to always randomly permute her responses would take extra time, and it may also require additional space if she needs to store every response query so that users can refer back to her responses for other, limited types of selection queries she may allow. So the technique of using random permutations can reduce the risks associated with the Mastermind cloning attack, but it doesn't necessarily eliminate these risks, and it comes with additional costs.

Acknowledgments

A shorter version of the material in this paper (which only dealt with binary attribute vectors) was presented by the authors at the ACM Workshop on Privacy in the Electronic Society (WPES), Chicago, October 2010. The authors would like to thank Pierre Baldi and Padhraic Smyth for, respectively, suggesting the privacy of genomic data and Facebook relationships as research topics. They are also grateful to Athina Markopoulou and Minas Gjokas for providing the Facebook-Uniform data. They would also like to acknowledge David Eppstein and Daniel Hirschberg for helpful discussions regarding the group testing topics of this paper. Furthermore, we would like to thank the reviewers and the editor for their helpful comments which have improved this paper. This research was supported by the US Office of Naval Research under MURI grant N00014-08-1-1015 and by the US National Science Foundation under grants 0724806, 0713046, 0847968, and a US National Science Foundation (NSF) Graduate Fellowship.

References

[1] P. Baldi, R.W. Benz, D. Hirschberg, and S. Swamidass, "Lossless Compression of Chemical Fingerprints Using Integer Entropy Codes Improves Storage and Retrieval," *J. Chemical Information and Modeling*, vol. 47, no. 6, pp. 2098-2109, 2007.

[2] S. Swamidass and P. Baldi, "Bounds and Algorithms for Exact Searches of Chemical Fingerprints in Linear and Sub-Linear Time," *J. Chemical Information and Modeling*, vol. 47, no. 2, pp. 302-317, 2007.

[3] A. Narayanan and V. Shmatikov, "De-Anonymizing Social Networks," *Proc. IEEE 30th Symp. Security and Privacy (SP '09)*, pp. 173-187, 2009.

[4] M.J. Atallah, F. Kerschbaum, and W. Du, "Secure and Private Sequence Comparisons," *Proc. ACM Workshop Privacy in the Electronic Soc. (WPES '03)*, pp. 39-44, 2003.

[5] A. Ben-David, N. Nisan, and B. Pinkas, "FairplayMP - A System for Secure Multi-Party Computation," *Proc. ACM Symp. Computer and Comm. Security (CCS)*, pp. 257-266, 2008.

[6] I. Damgård, M. Fitzi, E. Kiltz, J.B. Nielsen, and T. Toft, "Unconditionally Secure Constant-Rounds Multi-Party Computation for Equality, Comparison, Bits and Exponentiation," *Proc. Third Theory of Cryptography Conf.*, S. Halevi and T. Rabin, eds., pp. 285-304, 2006.

[7] S. Jha, L. Kruger, and V. Shmatikov, "Towards Practical Privacy for Genomic Computation," *Proc. IEEE Symp. Security and Privacy*, pp. 216-230, 2008.

[8] W. Jiang, M. Murugesan, C. Clifton, and L. Si, "Similar Document Detection with Limited Information Disclosure," *Proc. IEEE 24th Int'l Conf. Data Eng.*, pp. 735-743, 2008.

[9] D. Szajda, M. Pohl, J. Owen, and B.G. Lawson, "Toward a Practical Data Privacy Scheme for a Distributed Implementation of the Smith-Waterman Genome Sequence Comparison Algorithm," *Proc. Network and Distributed System Security (NDSS) Symp.*, 2006.

[10] Y. Sang and H. Shen, "Privacy Preserving Set Intersection Protocol Secure against Malicious Behaviors," *Proc. Eighth Int'l Conf. Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, pp. 461-468, 2007.

[11] A.C. Yao, "Protocols for Secure Computations," *Proc. 23rd Symp. Foundations of Computer Science (FOCS)*, pp. 160-164, 1982.

[12] D.S. Hirschberg and P. Baldi, "Effective Compression of Monotone and Quasi-Monotone Sequences of Integers," *Proc. Data Compression Conf. (DCC '08)*, 2008.

[13] B. Pakendorf and M. Stoneking, "Mitochondrial DNA and Human Evolution," *Ann. Rev. Genomics Human Genetics*, vol. 6, pp. 165-183, 2005.

[14] D.M. Behar1, S. Rosset, J. Blue-Smith, O. Balanovsky, S. Tzur1, D. Comas, R.J. Mitchell, L. Quintana-Murci, C. Tyler-Smith, and R.S. Wells, "The Genographic Project Public Participation Mitochondrial DNA Database," *PLoS Genetics*, vol. 3, no. 6, 2005.

[15] M. Brandon, M. Lott, K. Nguyen, S. Spolim, S. Navathe, P. Baldi, and D. Wallace, "MITOMAP: A Human Mitochondrial Genome Database - 2004 Update," *Nucleic Acids Research*, vol. 33, pp. 611-613, 2005.

[16] E. Ruiz-Pesini, M.T. Lott, V. Procaccio, J. Poole, M.C. Brandon, D. Mishmar, C. Yi, J. Kreuziger, P. Baldi, and D.C. Wallace, "An Enhanced MITOMAP with a Global mtDNA Mutational Philogeny," *Nucleic Acids Research*, vol. 35, pp. D823-D828, 2007.

[17] S. Harihara, M. Hirai, Y. Suutou, K. Shimizu, and K. Omoto, "Frequency of a 9-bp Deletion in the Mitochondrial DNA among Asian Populations," *Human Biology*, vol. 64, no. 2, pp. 161-166, 1992.

[18] K. Lewis, J. Kaufman, M. Gonzalez, A. Wimmer, and N. Christakis, "Tastes, Ties, and Time: A New Social Network Dataset Using Facebook.com," *Social Networks*, vol. 30, no. 4, pp. 330-342, 2008.

[19] R. Gross, A. Acquisti, and H.J. Heinz III, "Information Revelation and Privacy in Online Social Networks," *Proc. ACM Workshop Privacy in the Electronic Soc. (WPES '05)*, pp. 71-80, 2005.

[20] L.A. Stern and K. Taylor, "Social Networking on Facebook," *J. Comm., Speech & Theatre Assoc. of North Dakota,* vol. 20, pp. 9-20, 2007.

[21] M.T. Goodrich, "The Mastermind Attack on Genomic Data," *Proc. IEEE Symp. Security and Privacy,* pp. 204-218, 2009.

[22] V. Chvátal, "Mastermind," *Combinatorica,* vol. 3, nos. 3/4, pp. 325-329, 1983.

[23] D. Knuth, "The Computer as a Master Mind," *J. Recreational Math.,* vol. 9, pp. 1-5, 1977.

[24] Z. Chen, C. Cunha, and S. Homer, "Finding A Hidden Code by Asking Questions," *Proc. Second Int'l Conf. Computing and Combinatorics (COCOON),* pp. 50-55, 1996.

[25] M.T. Goodrich, "On the Algorithmic Complexity of the Mastermind Game with Black-Peg Results," *Information Processing Letters,* vol. 109, no. 13, pp. 675-678, 2009.

[26] F. Bancilhon and N. Spyratos, "Protection of Information in Relational Data Bases," *Proc. Third Int'l Conf. Very Large Data Bases (VLDB '77),* pp. 494-500, 1977.

[27] A. Deutsch and Y. Papakonstantinou, "Privacy in Database Publishing," *Proc. 10th Int'l Conf. Database Theory (ICDT),* T. Eiter and L. Libkin, eds., pp. 230-245, 2005.

[28] G. Miklau and D. Suciu, "A Formal Analysis of Information Disclosure in Data Exchange," *J. Computer and System Sciences,* vol. 73, no. 3, pp. 507-534, 2007.

[29] M. Kantarcioğlu, J. Jin, and C. Clifton, "When Do Data Mining Results Violate Privacy?," *Proc. 10th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '04),* pp. 599-604, 2004.

[30] R. Agrawal and J. Kiernan, "Watermarking Relational Databases," *Proc. 28th Int'l Conf. Very Large Data Bases (VLDB '02),* pp. 155-166, 2002.

[31] R. Agrawal, P.J. Haas, and J. Kiernan, "A System for Watermarking Relational Databases," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* p. 674, 2003.

[32] D. Gross-Amblard, "Query-Preserving Watermarking of Relational Databases and XML Documents," *Proc. 22nd ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems (PODS '03),* pp. 191-201, 2003.

[33] G. Schulz and M. Voigt, "A High Capacity Watermarking System for Digital Maps," *Proc. Workshop Multimedia and Security (MM&Sec '04),* pp. 180-186, 2004.

[34] R. Sion, M. Atallah, and S. Prabhakar, "Rights Protection for Relational Data," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 98-109, 2003.

[35] R. Sion, "Rights Assessment for Relational Data," *Proc. Secure Data Management in Decentralized Systems,* T. Yu and S. Jajodia, eds., pp. 427-457, Springer, 2007.

[36] K. LeFevre, D.J. Dewitt, and R. Ramakrishnan, "Incognito: Efficient Full-Domain K-Anonymity," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 49-60, 2005.

[37] P. Samarati, "Protecting Respondents' Identities in Microdata Release," *IEEE Trans. Knowledge and Data Eng.,* vol. 13, no. 6, pp. 1010-1027, Nov./Dec. 2001.

[38] P. Samarati and L. Sweeney, "Protecting Privacy when Disclosing Information: K-Anonymity and Its Enforcement through Generalization and Suppression," SRI, technical report, 1998.

[39] A. Meyerson and R. Williams, "On the Complexity of Optimal K-Anonymity," *Proc. 23rd ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems (PODS '04),* pp. 223-228, 2004.

[40] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu, "Anonymizing Tables," *Proc. 10th Int'l Conf. Database Theory (ICDT),* pp. 246-258, 2005.

[41] J.-W. Byun, A. Kamra, E. Bertino, and N. Li, "Efficient K-Anonymization Using Clustering Techniques," *Proc. 12th Int'l Conf. Database Systems for Advanced Applications (DASFAA),* pp. 188-200, 2007.

[42] S. Zhong, Z. Yang, and R.N. Wright, "Privacy-enhancing K-Anonymization of Customer Data," *Proc. 24th ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems (PODS '05),* pp. 139-147, 2005.

[43] R.J. Bayardo and R. Agrawal, "Data Privacy through Optimal K-Anonymization," *Proc. 21st Int'l Conf. Data Eng. (ICDE),* pp. 217-228, 2005.

[44] W. Du and M.J. Atallah, "Secure Multi-Party Computation Problems and Their Applications: A Review and Open Problems," *Proc. Workshop New Security Paradigms (NSPW),* pp. 13-22, 2001.

[45] M. Freedman, K. Nissim, and B. Pinkas, "Efficient Private Matching and Set Intersection," *Proc. EUROCRYPT Advances in Cryptology,* 2004.

[46] M.J. Atallah and J. Li, "Secure Outsourcing of Sequence Comparisons," *Int'l J. Information Security,* vol. 4, no. 4, pp. 277-287, 2005.

[47] D.S. Hirschberg, "A Linear Space Algorithm for Computing Maximal Common Subsequences," *Comm. ACM,* vol. 18, no. 6, pp. 341-343, 1975.

[48] C.S. Iliopoulos and M.S. Rahman, "Algorithms for Computing Variants of the Longest Common Subsequence Problem," *Theoretical Computer Science,* vol. 395, nos. 2/3, pp. 255-267, 2008.

[49] J.D. Ullman, A.V. Aho, and D.S. Hirschberg, "Bounds on the Complexity of the Longest Common Subsequence Problem," *J. ACM,* vol. 23, no. 1, pp. 1-12, 1976.

[50] J.R. Troncoso-Pastoriza, S. Katzenbeisser, and M. Celik, "Privacy Preserving Error Resilient DNA Searching through Oblivious Automata," *Proc. 14th ACM Conf. Computer and Comm. Security (CCS),* pp. 519-528, 2007.

[51] A. Amirbekyan and V. Estivill-Castro, "A New Efficient Privacy-Preserving Scalar Product Protocol," *Proc. Sixth Australasian Conf. Data Mining and Analytics (AusDM '07),* pp. 209-214, 2007.

[52] J. Vaidya and C. Clifton, "Secure Set Intersection Cardinality with Application to Association Rule Mining," *J. Computer Security,* vol. 13, no. 4, pp. 593-622, 2005.

[53] Y. Sang and H. Shen, "Privacy Preserving Set Intersection Based on Bilinear Groups," *Proc. 31st Australasian Conf. Computer Science (ACSC),* pp. 47-54, 2008.

[54] O. Goldreich, S. Micali, and A. Wigderson, "How to Play Any Mental Game," *Proc. 19th Ann. ACM Symp. Theory of Computing (STOC '87),* pp. 218-229, 1987.

[55] W. Du and M.J. Atallah, "Protocols for Secure Remote Database Access with Approximate Matching," *E-Commerce Security and Privacy: Advances in Information Security,* A.K. Ghosh, ed., vol. 2, pp. 87-112, Kluwer Academic Publishers, 2001.

[56] L. Backstrom, C. Dwork, and J. Kleinberg, "Wherefore Art Thou r3579x?: Anonymized Social Networks, Hidden Patterns, and Structural Steganography," *Proc. 16th Int'l Conf. World Wide Web (WWW '07),* pp. 181-190, 2007.

[57] A. Narayanan and V. Shmatikov, "Robust De-anonymization of Large Sparse Datasets," *Proc. IEEE Symp. Security and Privacy (SP '08),* pp. 111-125, 2008.

[58] R. Dorfman, "The Detection of Defective Members of Large Populations," *Annals of Math. Statistics,* vol. 14, pp. 436-440, 1943.

[59] D.-Z. Du and F.K. Hwang, *Combinatorial Group Testing and Its Applications,* second ed. World Scientific, 2000.

[60] D. Eppstein, M.T. Goodrich, and D.S. Hirschberg, "Improved Combinatorial Group Testing for Real-World Problem Sizes," *Proc. Workshop Algorithms and Data Structures (WADS),* 2005.

[61] M. Ruszinkó, "On the Upper Bound of the Size of the r-Cover-Free Families," *J. Combinatorial Theory Series A,* vol. 66, pp. 302-310, 1994.

[62] D. Knuth, *The Art of Computer Programming.* Addison-Wesley, 1973.

[63] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Signed Networks in Social Media," *Proc. 28th ACM Conf. Human Factors in Computing Systems,* 2010.

[64] M. Gjoka, M. Kurant, C. Butts, and A. Markopoulou, "Walking in Facebook: A Case Study of Unbiased Sampling of OSNs," *Proc. IEEE INFOCOM,* pp. 1-9, 2010.

[65] A.L. Traud, E.D. Kelsic, P.J. Mucha, and M.A. Porter, "Community Structure in Online Collegiate Social Networks," arXiv:0809.0960, 2008.

[66] A.U. Asuncion and M.T. Goodrich, "Turning Privacy Leaks into Floods: Surreptitious Discovery of Social Network Friendships and Other Sensitive Binary Attribute Vectors," *Proc. Ninth Ann. ACM Workshop Privacy in the Electronic Soc. (WPES '10),* pp. 21-30, 2010.

Arthur U. Asuncion is working toward the PhD degree in the Donald Bren School of Information and Computer Sciences at the University of California, Irvine. His primary research contributions are to artificial intelligence and statistical machine learning; furthermore, he also has interests in distributed computing, privacy preservation, and social network analysis. He is a US National Science Foundation (NSF) Graduate Fellow and a recent Yahoo! Key Scientific Challenges Program Winner.

Michael T. Goodrich is a chancellor's professor in the Donald Bren School of Information and Computer Sciences at the University of California, Irvine. With more than 200 publications, including several widely adopted books, his recent work includes contributions to efficient and secure distributed data structures, information privacy, social networks, and network/grid security. He is an ACM Distinguished scientist, a fellow of the American Association for the Advancement of Science (AAAS), a Fulbright Scholar, a fellow of the IEEE, a fellow of the ACM, and a recipient of numerous research awards.