Turning Privacy Leaks into Floods: Surreptitious Discovery of Social Network Friendships and Other Sensitive Binary Attribute Vectors

Arthur U. Asuncion
Department of Computer Science
University of California, Irvine
Irvine, CA 92697 USA
asuncion@ics.uci.edu

Michael T. Goodrich
Department of Computer Science
University of California, Irvine
Irvine, CA 92697 USA
goodrich@ics.uci.edu

ABSTRACT

We study methods for attacking the privacy of social networking sites, collaborative filtering sites, databases of genetic signatures, and other data sets that can be represented as vectors of binary relationships. Our methods are based on reductions to nonadaptive group testing, which implies that our methods can exploit a minimal amount of privacy leakage, such as contained in a single bit that indicates if two people in a social network have a friend in common or not. We analyze our methods for turning such privacy leaks into floods using theoretical characterizations as well as experimental tests. Our empirical analyses are based on experiments involving privacy attacks on the social networking sites Facebook and LiveJournal, a database of mitochondrial DNA, a power grid network, and the movie-ratings database released as a part of the Netflix Prize contest. For instance, with respect to Facebook, our analysis shows that it is effectively possible to break the privacy of members who restrict their friends lists to friends-of-friends.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems; K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*

General Terms

Security

Keywords

Social networks, privacy leaks, genetic signatures, binary attribute vectors, combinatorial group testing

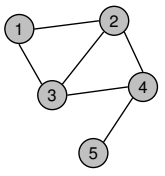
1. INTRODUCTION

Each time a website answers a query or displays information related to its users, it leaks a little piece of itself in its response. Such a privacy leak could be as small as a single bit, which might at first not be cause for much concern, or a complete disclosure about one of the users of that website. For instance, any social networking site interested in growing its user base will likely supply some method for identifying mutual friends between two individuals, which, for the sake of this paper, we are willing to reduce to a single bit that indicates whether two individuals have a mutual friend in common or not. This is arguably the minimal amount of mutual friendship information that any commercially-successful social network would ever leak (and most leak much more than this amount). Unfortunately, as we show in this paper, this minimal amount of privacy leakage is enough to allow for the revelation of all the information a site is trying to protect by limiting such query responses. Using both theoretical and empirical analyses, we demonstrate that nonadaptive attacks are able to clone real-world databases in a very efficient manner by exploiting the sparsity in the data.

1.1 Binary Attribute Vectors

We focus on information representable as *binary attribute* vectors. Such a vector, e.g., $v = (0, 0, 0, 1, 0, 1, 1, \ldots, 0)$, reflects the presence or absence of each of a large number of possible characteristics, with each bit representing a single potential binary relationship. Examples of such vectors include the following:

- Each row in an adjacency matrix for part of a social network, such as Facebook, is a binary vector that encodes friendships for an individual in that subnetwork. (See Figure 1.)
- In a *genetic signature* vector, a binary vector can be defined from the presence or absence of mutations in a DNA string.
- In a collaborative filtering ratings vector, as used by companies such as Amazon and Netflix, a binary vector can be defined so that each position represents a product, and the corresponding bit is 1 if and only if the associated person rated that product above some threshold.



	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

Figure 1: An example graph and its adjacency matrix.

In the context of online data, binary attribute vectors have obvious privacy considerations. For instance, Bob's genetic signature could be used by an unethical employer or insurance company to discriminate against him based on his risks for future diseases. Also, it is possible using a genetic signature derived from a short

WPES'10, October 4, 2010, Chicago, Illinois, USA.

string of Bob's mitochondrial DNA to trace his maternal lineage to an ancestral location [6, 31], which is information that could then be used for ethnic discrimination [21]. Likewise, Bob's movie ratings could be used to infer personal information, such as political or religious orientation. In addition, knowing Alice's set of friends is a gateway privacy leak, for friendship overlaps have been shown to be sufficient to de-anonymize individuals across multiple social networking sites [30] and has even been identified as a possible indicator of sexual orientation [22]. Alice might not be worried about showing her true identity in Facebook, where her status updates are always about songs and videos, but she might have concerns if her true identity in a pro-democracy social networking site is discovered by the repressive regime that runs her country.

1.2 The Group-Testing Attack

Group testing was introduced by Dorfman [9], 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 something we are calling the *group-testing attack*.

Formally, in a group testing scenario [10], 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 query from any subset T of S and, in a single *test*, determine if T contains any defective items or not. A group testing regimen is a set, M, of tests and an algorithm for determining the identities of the defective items based on the outcomes of the tests in M.

In a *group-testing attack*, a *querier*, Bob, is allowed certain types of queries to a collection, $\mathcal{X} = (X_1, X_2, \ldots, X_g)$, of binary attribute vectors. Bob's goal is to efficiently replicate as many vectors in $\mathcal{X}$ as possible through a small number of queries. With respect to the types of queries Bob is allowed, we assume that the collection supports *intersection* queries from Bob. In an intersection query, Bob provides a single string or vector Q and in $\mathcal{X}$'s response he receives a binary response vector $R = (r_1, r_2, \ldots, r_g)$, where each r_i is 1 if and only if there is a bit position that is 1 in both X_i and Q, i.e., $r_i = 0$ only if $X_i \wedge Q = \vec{0}$, where $\wedge$ denotes bitwise AND. Moreover, we assume that Bob learns nothing more than this, either because the website for $\mathcal{X}$ provides only this information or because Bob and $\mathcal{X}$'s owner engaged in an Secure Multiparty Computation (SMC) protocol (e.g., see [1, 16, 18, 34, 37–39]) to determine R so that no other information is revealed.

If, in practice, Bob learns more than the information contained in R, that only strengthens his attack. The point of this paper is that even with just the information leaked in R, Bob can construct a small number of query vectors, Q_1, Q_2, ..., Q_k, that are sufficient to learn all or a sizeable fraction of the vectors in $\mathcal{X}$, which amounts to a flood of information. Moreover, our group-testing 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).

1.3 Attack Scenarios

We have deliberately described the group-testing attack in fairly abstract terms, so as to show how it applies to a wide variety of attack scenarios. We outline three such attack scenarios below.

1.3.1 Social Networking Sites

Suppose the vectors in $\mathcal{X}$ represent the rows of the adjacency matrix defined by the friendship ties for a social networking site, 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 [27] or do a cross-networking identification attack [30], since 89% of Facebook users use their real names [20]. In this case, Bob's query vectors correspond to a relatively small number of pseudonyms that Bob creates 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 [35]). 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 mutual 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.

1.3.2 Collaborative Filtering Sites

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 1 in position j if and only if person i rates product j above some minimum threshold. 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, we 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 share any ratings in common with users in the target population. So long as the collaborative filtering web site allows for users to check for overlapping scores with other users, Bob can employ the group-testing attack.

1.3.3 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. That is, we number all the known genetic mutations with

respect to a reference DNA string, R, and each vector X_i is associated with an individual, who we will call Alice, such that there is a 1 in position j of X_i if and only if Alice's DNA has mutation j with respect to R. For example, in mitochondrial DNA, the reference R is roughly 16,500 base pairs long, but has only about 4,000 known mutations [7, 32]. Bob's goal in this group-testing 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 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. For 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 $\mathcal{X}$.

Realistic attacks can also be constructed in other domains. Sensitive image data, such as captured by biometric devices, may be represented as sparse binary vectors, making it susceptible to a group-testing attack, especially when efficient tools exist for comparing a query (e.g. a fingerprint or an iris scan) to the entire database.

1.4 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 just a relatively small number of single-bit privacy leaks. The risk to the group-testing attack can therefore be characterized in terms of 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 $\mathcal{X}$. The critical factor here is thus a parameter, d, which, in the group testing context, refers to the small number of "defective" items in the large group, such as the few thousands of syphilis-infected blood samples out of the millions donated during World War II.

Interestingly, each of the attack scenarios mentioned above possess such a parameter, allowing for Bob to employ efficient group-testing attacks with a relatively small number of queries. For example, most people in social networking sites, such as Facebook, have less than a few hundred friends (as we show below), which implies that the degree distributions of friendship in such networks don't closely follow a power law. Likewise, most collaborative filtering preference vectors, such as in the Netflix Prize contest, have ratings for at most a few hundred items. Similarly, an individual's genetic signature will typically have a relatively small number of indicators for mutations with respect to a reference DNA string, R. For example, with mitochondrial DNA, most people have fewer than 100 mutations with respect to a commonly-used reference string. Thus, there are several modern contexts that have all the pieces in place to allow for the group-testing attack to be used.

1.5 Related Work

Following a framework by Bancilhon and Spyratos [5], Deutsch and Papakonstantinou [8] 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 specific secret, $\mathcal{S}$, that the data owner, Alice, is trying to protect. Attackers are allowed to form legal queries and ask them of the database, while Alice tries to protect the information that these queries leak about the secret $\mathcal{S}$. Note that this framework is related to the group-testing attack, but these two are not identical, since in the group-testing attack there is no specifically sensitive part of the data. Instead, Bob has a quantitative goal of learning as much of the database $\mathcal{X}$ as possible. Similarly, Kantarcioğlu *et al.* [25] study models that quantify the degree to which data mining searches expose private information, but this model is also not directly applicable to the group-testing attack.

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. Such protocols have been developed for comparisons done with genomic sequences (e.g., see [2, 12, 16]). In particular, Atallah *et al.* [2] and Atallah and Li [3] studied privacy-preserving protocols for edit-distance sequence comparisons. Troncoso-Pastoriza *et al.* [37] described a privacy-preserving protocol for regular-expression searching in a DNA sequence. Jha *et al.* [23] give privacy-preserving protocols for computing edit distance and Smith-Waterman similarity scores between two genomic sequences. Aligned matching results between two binary vectors can be done in a privacy-preserving manner, as well, using privacy-preserving set intersection protocols (e.g., see [1, 16, 34, 38]). Du and Atallah [11] 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. 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 consider in this paper. The SMC method of Jiang *et al.* [24], on the other hand, is directly applicable. It provides a vector of scores comparing a vector Q to a sequence of vectors, as we require in this paper. Thus, the group-testing attack can be viewed as an attack on repeated use of the SMC protocol of Jiang *et al.*

Goodrich [19] studies the problem of discovering a single DNA string from a series of genomic comparison queries. All of his methods are sequential and adaptive, however, so they are not applicable to group-testing attack scenarios, as outlined in this paper, that require oblivious, nonadaptive sets of queries. Others have investigated de-anonymization techniques on social networks [4] and Netflix data [29]; these works are complementary to our goal of cloning the databases. Dwork *et al.* [14] studied the task of recovering statistical databases using sum queries and LP decoding. While these techniques can be potentially useful to our cause, we consider the more restrictive case of receiving only binary responses, and we show that these bit-sized privacy leaks are enough to efficiently clone a variety of real-world databases.

1.6 Our Results

We present a number of algorithms for performing group-testing attacks on an entire database, $\mathcal{X} = (X_1, X_2, \ldots, X_g)$, of binary attribute vectors, so as to replicate all or a large portion of $\mathcal{X}$. All of our methods assume only a minimal amount of information leakage per query, where a querier, Bob, issues a binary query vector, Q, and receives a vector of responses $(r_1, r_2, \ldots, r_g)$, where each r_i is a single bit indicating if Q and X_i share a 1 in the same position or not. We assume that all the strings in $\mathcal{X}$ are the same length, since we can view smaller strings as being padded with zeroes. Specifically, we show the following:

- Suppose $\mathcal{X}$ contains g binary vectors, each of length n, with at least $g' \leq g$ of these vectors having at most $d < n$ differences from a public reference vector. Then, with high probability, at least g' of the vectors in $\mathcal{X}$ can be completely determined using a number of queries that is at most

$$4d \log n + 2 \min\{d \log g, d^2 \log(en/d)\}.$$

The efficient attack algorithms based on the proof of this fact use sparsity to exploit the property that binary attribute vectors in many real-world scenarios can be characterized in terms of a small number of differences with a reference string, R. Thus, our main contribution is the development of an efficient privacy attack that is based on randomized nonadaptive group-testing (which uses only binary responses) and that is applicable to many real-world databases. Our randomized construction provides surreptitious queries, and our use of nonadaptive techniques allows for queries to be run in parallel on large databases.

Another contribution of this paper is an empirical analysis of our proposed attack on various real-world databases, as well as algorithmic heuristics for further exploiting sparsity. We provide experimental results showing that nonadaptive group-testing attacks can work effectively in various contexts, as we apply it to friendship discovery in subsets of the Facebook and LiveJournal networks, a database of mitochondrial DNA (mtDNA) strings, a power grid network, and the database of movie-ratings vectors provided for the Netflix Prize contest. In each case, our experiments show that large portions of these data sets can be replicated using a number of queries that is much smaller than the length of the vectors in these databases. We also demonstrate how to take further advantage of the sparsity of the data by empirically using a parameter $\hat{d}$ in our algorithms that is much smaller than d. For instance, using a small $\hat{d}$, we are able to clone half of the LiveJournal database (where each vector has 5 million entries), in as few as 300 queries.

2. NONADAPTIVE GROUP TESTING

As mentioned above, in the *combinatorial group testing* problem (e.g., see Du and Hwang [10]), 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 *oblivious* or *nonadaptive*. For the application to group-testing attacks, we are interested in nonadaptive methods.

There are several existing nonadaptive group testing methods (e.g., see Du and Hwang [10]), but these approaches are meant for a more general context than applies in the group-testing 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 vectors we are interested in determining. Because of this, we can, in fact, derive improved bounds than would be implied by existing combinatorial group-testing methods.

Suppose, then, 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 an adaptation of a randomized approach used by Eppstein *et al.* [15].

A nonadaptive group testing algorithm can actually be viewed as a $t \times n$ 0-1 matrix, M. Each of the n columns of M corresponds to one of the n items and each of the t 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 large t must be for the results of these tests 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 [10], 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, 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 [10, 33]. 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.

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 for columns that correspond to samples that are distinguishable from the defectives 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 the 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 non-defective 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 non-zero 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, 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 [26]) 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 $\mathcal{C}$.

3. ATTACK CONSTRUCTION

In this section, we describe how to use nonadaptive group testing to construct an efficient group-testing attack. Suppose we are given a database $\mathcal{X}$ of g binary attribute vectors,

$$\mathcal{X} = \{X_1, X_2, \ldots, X_g\},$$

each of length n, for which Bob is allowed to perform comparison queries.

3.1 The Standard Group-Testing Attack

In the standard group-testing attack, we assume that Bob makes aligned match queries, where Q and each X_i are assumed to have length exactly n and the score between a query string Q and each string X_i in $\mathcal{X}$ is measured by the response

$$r(Q, X_i) = \begin{cases} 0 & \text{if } Q \wedge X_i = \vec{0} \\ 1 & \text{else.} \end{cases}$$

Thus, interpreting Q as a group test, this bit, $r(Q, X_i)$ is 1 if and only if the group test for Q is "negative," that is, it detects a "defective" item in the group defined by the positions in Q that have bit values equal to 1. So, the attack for Bob is perform the set of tests defined by the rows of a random matrix, M, that is defined so that, with high probability, M is $\mathcal{D}$-distinguished for a collection, $\mathcal{D} = \{D_1, D_2, \ldots, D_g\}$, where D_i is the set of bit positions in X_i where X_i has a 1. Thus, we have the following.

Theorem 2: *There is a nonadaptive group-testing attack, which can discover each of the g binary attribute vector in $\mathcal{X}$, using $2t$ tests, with probability at least $1 - 1/n$, 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 ones in any vector in $\mathcal{X}$.

Suppose, for example, that Bob knows, by Theorems 1 and 2, that a random M having t rows, as defined above, is $\mathcal{D}$-distinguished with high probability. Then if he is doing a group-testing attack on a social networking site, he needs to create t pseudonyms in the social network, give each of them a random set of friends using a random process that defines M as described above, and then use each of the pseudonyms to check if they have any friends in common with a target victim, X_i. The results of these tests will be sufficient, with high probability, to identify all the friends of X_i, even if he or she has restricted mutual friendship disclosure to only be allowed for "friends of friends," as is a standard privacy setting in Facebook.

3.2 Extensions to the Group-Testing Attack

In some cases, such as in contexts with attacks on genetic signatures, the queries Bob gets to employ don't actually return a bit that indicates whether there are any overlaps with Bob's query vector. Instead, genetic SMC computations (e.g., see [1,16,34,37,38]) typically return a score indicating the edit distance between a query DNA string or genetic signature vector, Q, and the subject string or vector, X_i. In these cases, the response Bob gets from his query is equivalent to the score,

$$r(Q, X_i) = |\{j \colon Q[j] = X_i[j]\}|.$$

In addition, it is also common in these cases that there is a reference string or vector, R, such that there is a parameter $d \leq n$ so that each string in $\mathcal{X}$ is known to have at most d differences with a public reference string R. In the description of the group-testing attack given above, we have implicitly assumed that $R = \vec{0}$, that is, that the reference Bob is testing against is a vector of all zeroes. We can extend the group-testing attack to other scenarios, however, as is common in genetic applications, where the reference string is not all zeroes. For example, certain mutations might be so common in a population that Bob knows he should assume such mutations are included in his reference string, R. As we explore in our experiments, in Section 4, such a string R and parameter d are not uncommon in biological applications and d is often much smaller than average number of ones in a genetic signature.

So, let us describe how to extend the group-testing attack to these more general settings. Let us start by assuming we have a $2t \times n$ nonadaptive group testing matrix, M, for a set of size n having at most d defectives, where

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

We begin our group-testing attack, in this case, by making a query for the reference string, R. Let r be the response score for the query

for R. Next, we create a different string query Q_k for each of the $2t$ tests in M, defined as follows:

$$Q_k[j] = \begin{cases} R[j] & \text{if } M[k,j] = 0 \\ (R[j]+1) \bmod 2 & \text{else.} \end{cases}$$

This query will have some response, r_k. We interpret test k as having a "positive" response, that is, it does not detect a defective item, if

$$r_k = r - b_k,$$

where b_k is the number of 1's in row k of M. Intuitively, each 1 in row k of M indicates a place where we test for a deviation from R from its reference value at that location to the alternative bit. If none of these locations is a match with the current X_i string, then all these locations take their reference values. In other words, defective "items" in the associated group testing method correspond to locations where X_i differs from the reference string. And since there is only one other choice for the alternative character, differences from the reference string have to be matches for the complementary bit. 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 3: *There is a nonadaptive group-testing attack, which can discover each of the g binary strings in $\mathcal{X}$, using $2t$ tests, with probability at least $1 - 1/n$, 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 a reference string R, even if each response is a difference score rather than a bit indicating nonempty intersections.

4. EXPERIMENTAL ANALYSIS

To test the real-world risks of the group-testing attack, we applied our methods to online social networking sites, human mitochondrial DNA strings, power grid data, and movie ratings vectors taken from the Netflix Prize contest. We describe the data and present experimental results which demonstrate the effectiveness of our approach.

4.1 Data Sets

We use various real-world data sets, each with different characteristics. For each data set, Table 1 lists the number of vectors, vector length, and maximum difference from the reference string.

The social networks that we attack are Facebook and LiveJournal. In particular, we attack the adjacency matrices of the friend networks of each site. We use two different Facebook data sets: Facebook-Uniform and Facebook-UNC. Facebook-Uniform, provided by the authors of [17], 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 [36]. The LiveJournal data is a social network with approximately 5 million users[1]; in our experiments, we use a representative random subset of 1,000 users.

Table 1: Data sets used in experiments

Name	Vectors (g)	Length (n)	Max Diff (d)
Facebook-Uniform	1,000	72,261,577	2,164
Facebook-UNC	18,163	18,163	3,795
LiveJournal	1,000	4,847,571	320
Genomic	1,000	16,586	107
Power Grid	4,941	4,941	19
Netflix	1,000	17,770	4,395
Netflix-All	100,480,507	17,770	17,653

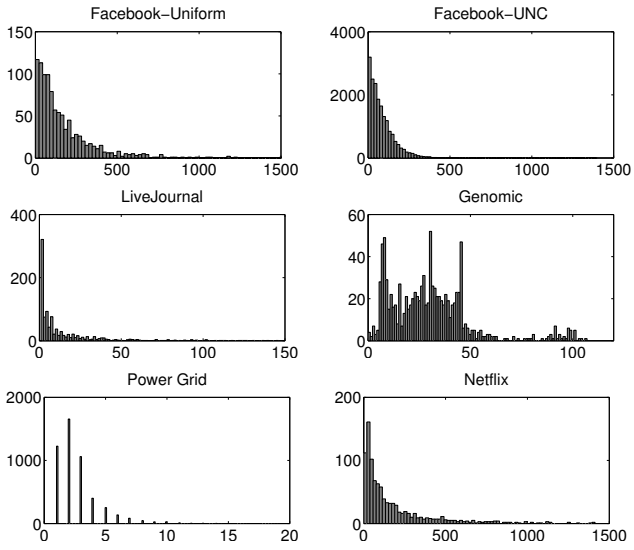


Figure 2: Histogram of distances from reference R, for each data set.

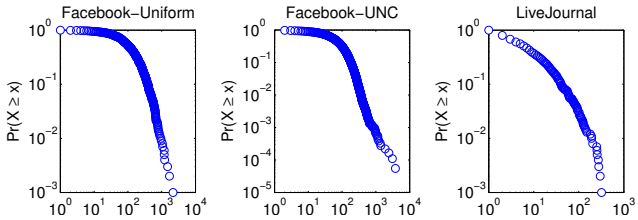


Figure 3: Complementary CDF of distances from R, on a log-log scale, for the social network data sets.

Due to the sparse nature of these networks, the reference string R we use in our attacks is a 0-vector. The distribution of differences from R (which is just the "degree distribution" in the case of social networks) is displayed in Figure 2. The complementary cumulative distribution function (CCDF) is shown in Figure 3. Power-law behavior would be exhibited as a line on this log-log scale. In the plots, we see that the curves are "bowed-up" which suggests that there is not as much mass in the tails of the distributions. This sparsity allows for efficient group-testing attacks, as we will see in the experimental results.

Our Genomic database consists of 1,000 human mitochondrial sequences downloadable from GenBank[2]. We use the Revised Cambridge Reference Sequence (rCRS), of length 16,586 bp, as the ref-

[1] http://snap.stanford.edu/data/
[2] http://www.ncbi.nlm.nih.gov/Genbank/

Table 2: Number of tests needed to clone (a) entire database according to theory, using d; (b) 50% of database, using d_{median}; (c) entire database, using baseline method.

	Theory (using d)	Theory (using d_{median})	Baseline method
Facebook-Uniform	272,664	13,608	72,261,577
Facebook-UNC	326,370	5,332	18,163
LiveJournal	35,200	660	4,847,571
Genomic	8,346	2,184	16,586
Power Grid	1,406	148	4,941
Netflix	342,810	7,332	17,770
Netflix-All	1,941,830	10,560	17,770

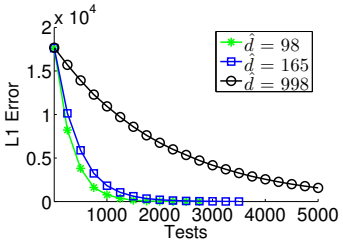


Figure 5: Error as a function of the number of tests for a single Netflix user, for various $\hat{d}$.

erence string R. All of the mtDNA sequences were aligned to R, and each sequence was encoded in binary fashion in terms of the locations in R where each string differed from the reference. Figure 2 shows the distribution of sequence differences from R.

The Power Grid data is an adjacency matrix of 4,941 nodes, representing the power grid network topology of the western United States. This data was originally compiled by Watts and Strogatz and is available online [13]. One can imagine a scenario where the topology of a power grid is maintained electronically in a database, making it potentially susceptible to a group-testing attack.

Our movie-rating data is taken from the Netflix Prize database[3], which consists of over 100 million movie ratings by 480,189 Netflix users. In our experiments, we mainly use a representative subset of 1,000 users. Exceptions, however, are the experiments in Figures 7 and Figure 8, in which we attack all 480,189 users. Each user has an associated vector defined over 17,770 movies, denoting the movies that the user rated, as well as the actual ratings (from 1 to 5) given by the user. For simplicity, our database only keeps track of whether or not a user rated a particular movie. Note that even these binary indicators can be highly informative of the user's preferences. Thus, each user is represented as a binary string of length 17,770. Our reference string in this case consists of all zeros, representing the case where no movies are rated. According to the Netflix distribution in Figure 2, the majority of users rate less than 300 movies, which suggests that this data set is sparse as well.

4.2 Experiments

We perform group-testing attacks on each data set in the following manner. Similar to randomly selecting $\frac{t}{d}$ rows from $2t$ rows (for each column in the nonadaptive group matrix M), we take a stochastic approach and set each entry in M to 1 with probability $p = \frac{1}{2d}$. This approach allows us to easily add additional tests to M until the vector is cloned. In our simulation of a cloning attack, for each vector in the database, additional tests are continually performed until the vector is exactly cloned or until a cutoff is reached (10,000 tests for Genomic, 20,000 tests for Power Grid, and 100,000 tests for the other data sets). Note that our attack is nonadaptive, since the construction of each test is independent of the results of previous tests. We initialize with the same random seed for each vector, ensuring that the same exact tests are performed on each vector. This setup allows us to determine the actual number of tests needed to clone the vectors.

Before presenting our empirical results, we display in Table 2 the number of tests needed to guarantee that the entire database is exactly cloned (with probability $1-1/n$), according to our theoretical bounds in Section 3. The number of tests is a function of d, the maximum number of differences from R across the entire database of vectors. If we are only interested in guaranteeing that half of the database is exactly cloned (namely the sparsest 50%), we can use the median number of differences from R, d_{median}, which significantly reduces the number of tests required. Furthermore, there is a simple baseline technique for cloning an entire database which requires exactly g tests, where g is the length of the vectors in the database. Each test Q_i, for $1 \leq i \leq g$, would simply be a vector with a 1 in index i and zeros elsewhere. In comparison to the baseline method, our theoretical bound (using d) is quite loose in the case of Facebook-UNC and Netflix, since the ratio between d and n is high for those data sets. However, for Facebook-Uniform, LiveJournal, Genomic, and Power Grid, our theoretical bound requires significantly less tests than the baseline method.

Since each vector's distance from R is usually much smaller than d, it is empirically advantageous to use a target $\hat{d}$ that is smaller than d. For the Facebook-UNC data, the maximum difference from R is 3,795 while the mean difference is 84 and the median is 62. Likewise, for the Genomic data, the maximum difference from R is 107, while the mean difference is 30 and the median is 28. Thus, there are different possible settings for $\hat{d}$, and some settings will allow us to significantly outperform both the baseline method and our theoretical bounds.

Figure 4 shows the average number of tests required (across all vectors in the database) until the vector is exactly cloned, as a function of $\hat{d}$. We exclude Facebook-Uniform and Netflix-All from this set of experiments due to their high dimensionality. In a few cases, when vectors are far from R, the algorithm may reach the cutoff value for the maximum number of tests, which may cause the mean to be undervalued; thus, we also plot the median number of tests since the median is more robust against outliers. For all these data sets, we see that the group-testing attack performs significantly better when $\hat{d}$ is significantly less than the maximum difference d. For instance, the average number of tests required to clone a LiveJournal vector, when $\hat{d} = d = 320$, is over 10,000; in comparison, when $\hat{d} = 17$ (which is actually the mean difference from R), the average number of required tests is 2,503 and the median is 641 tests, which is an order of magnitude improvement over using d. Likewise, we see that the optimal $\hat{d}$ for the Genomic database is around 22, while the optimal $\hat{d}$ for the Netflix data is around 165, when considering the mean curve. For each data set, the empirical median number of tests in Figure 4 compares favorably to the theoretical number of tests needed to guarantee that 50% of the database is cloned. For instance, when setting $\hat{d} = d_{\text{median}} = 62$ in Facebook-UNC, the median number of tests is 2,201, which is half the number of tests theoretically required. Likewise, when $\hat{d} = d_{\text{median}} = 94$ for Netflix, the median number of tests is around 3,200, which is significantly less than the theoretical bound.

[3] http://www.netflixprize.com

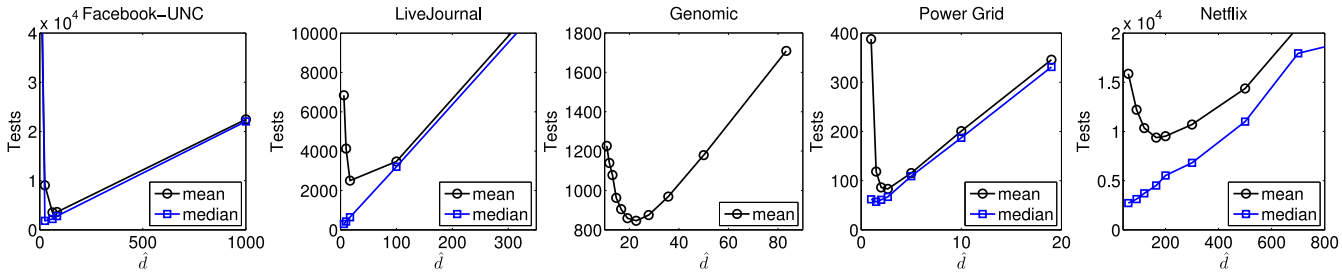


Figure 4: Mean (and median) number of tests required until a vector is cloned for various settings of target distance $\hat{d}$.

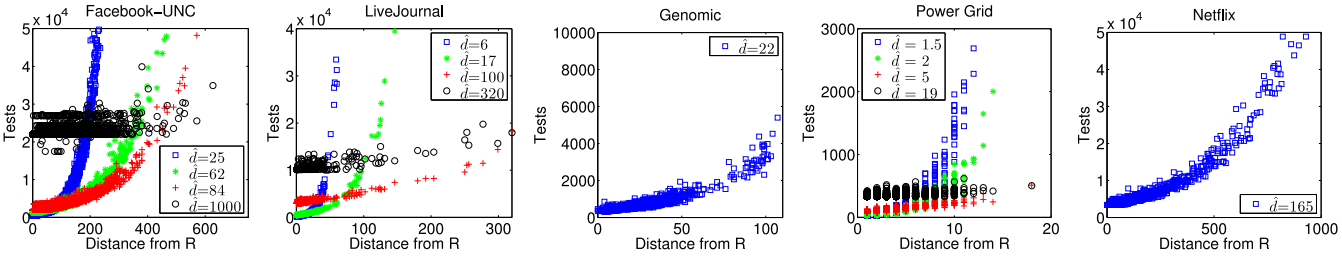


Figure 6: Number of tests required to clone each vector, ordered by distance from R. Each vector is a different point.

These results suggest that it is preferable in practice to use a $\hat{d}$ value that is significantly less than d and closer to the mean distance from R. These results also indicate a tradeoff. If $\hat{d}$ is too small, the number of ones in each test would increase and it would take longer to exactly clone a string that is far from R. If $\hat{d}$ is too large (e.g. $\hat{d} = d$), many inefficient tests would be performed on strings that are close to R. Figure 5 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 98 movies. One can see that using $\hat{d} = 998$ requires many more tests than using smaller settings for $\hat{d}$. We assume that a good estimate for $\hat{d}$ (such as the mean distance from R) can be obtained a priori, e.g., through scientific knowledge (in the case of Genomic data) or through public information (in the cases of Facebook, LiveJournal, and Netflix).

We also investigate the relationship between the number of required tests and the vector's distance from R. In Figure 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 the Facebook-UNC, LiveJournal, and Power Grid data, we display different scatter plots for different settings of $\hat{d}$. For instance, for the LiveJournal data, the number of tests is relatively constant across all distances when the $\hat{d} = d = 320$; 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 $\hat{d} = 17$, the number of required tests is only in the hundreds, around the vicinity 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.

In our final set of results, we show the percentage of the data set that is successfully attacked as a function of the number of tests, in Figure 7. For Facebook-UNC, we see that the group-testing attack displays different behavior for different choices of $\hat{d}$. When $\hat{d} = 3$, the attack is able to quickly recover (the sparsest) 15% 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% of the database has been successfully recovered after 2000 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 ran the same experiment on Facebook-Uniform for $\hat{d} = 108$ (the median distance from R) and see that over 70% of the data set can be reconstructed with 10,000 tests. 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% of a 300-million user Facebook network is cloned is only 22,000 (assuming $d_{\text{median}} = 130$)[4]. Since our attack usually outperforms the theoretical bounds, 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 queries.

The LiveJournal results in Figure 7 also show the effectiveness of the group-testing attack. Due to the sparsity of this data set, the algorithm is able to recover at least 90% of the database for a wide variety of settings for $\hat{d}$ after several thousands of tests. As in earlier plots, we see that the benefit of using $\hat{d} < d$ is the faster rate of progress. When $\hat{d} = d = 320$, no vectors are cloned until after 10,000 tests; in contrast, when $\hat{d} = 6$, 50% of the database is cloned after only 300 tests. Thus, while our theoretical results provide some nice probabilistic guarantees, our algorithm can perform better when $\hat{d}$ is significantly less than d, as long as the data set is sparse, which is often the case in practice.

[4] `http://www.facebook.com/press/info.php?statistics` reveals that $d_{\text{mean}} = 130$, and so d_{median} should be even smaller.

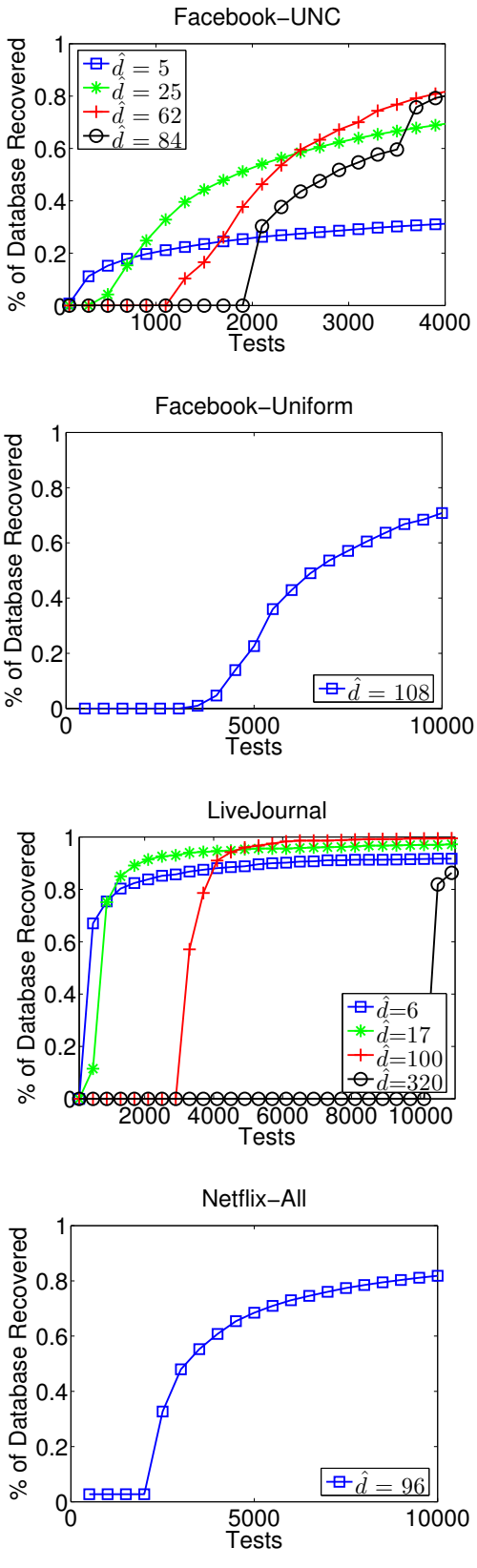


Figure 7: Percentage of data vectors successfully attacked, as a function of the number of nonadaptive tests.

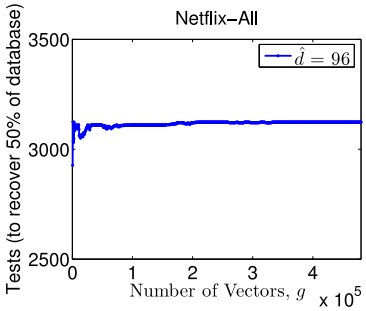


Figure 8: Number of tests needed to clone 50% of Netflix-All, as a function of the number of vectors g in that database.

Finally, we performed a coordinated Mastermind attack against the Netflix-All database (by running the same tests on all 480,189 users in parallel). We set $\hat{d} = 96$, the mean difference from R across the entire database. As shown in Figure 7, we can exactly clone 30% of the users with just 2,500 queries. After 5,000 tests, 65% of the users have been cloned, and nearly 80% of all Netflix users are cloned after 10,000 tests. In Figure 8, we show the number of tests needed to recover 50% of the Netflix-All database, as a function of the number of vectors, g, in that database. While our theoretical guarantees require that the number of tests scale with $\log(g)$, our empirical results for Netflix-All show that the number of tests needed stays nearly constant (at around 3,100 tests), even when we vary g from 500 to 480,189. These results suggest that our group-testing attack is very scalable. Even in a nonadaptive setting, it is possible to reconstruct over half of a database of size 480,189 $\times$ 17,770 with a few thousand tests.

In addition to the experiments in this section, we also performed experiments on other data, including the Facebook networks for Princeton and the University of Oklahoma [36]. We observed that the group-testing attack also performed well in these other settings.

Our empirical results have also shown that there is sensitivity to the choice of $\hat{d}$ in certain cases. One possible improvement to combat this sensitivity is to use a tiered approach, where $\hat{d}_1$ is used to construct the first 5000 tests, $\hat{d}_2$ is used to construct the next 5000 tests, etc. Each $\hat{d}_i$ could correspond to a mode in the database's distribution of differences from R. Nonetheless, even when using a single $\hat{d}$, it is possible to clone a large fraction of the database by simply performing a nonadaptive group-testing attack.

5. CONCLUSION

We have studied the group-testing attack, both from a theoretical and experimental perspective, and have shown its effectiveness in being able to copy the contents of a database of binary attribute vectors through a sequence of comparison queries. A natural direction for future work, of course, is on methods for defeating it, which we have not addressed in this paper. Certainly, if there is some way that the correspondence of query responses and query subjects could be randomly permuted, then that would help, since it would make it harder (but not necessarily impossible) for Bob to correlate responses between different queries. Of course, requiring $\mathcal{X}$ to always randomly permute its responses would take extra time, and it may also require additional space if we 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 group-testing attack, but it doesn't necessarily eliminate these risks, and it comes with additional costs.

Acknowledgments

We thank Pierre Baldi and Padhraic Smyth for suggesting the privacy of genomic data and Facebook friendships as research questions, and we thank Athina Markopoulou and Minas Gjokas for providing unbiased Facebook data. We also thank David Eppstein, and Daniel Hirschberg for discussions regarding group testing. This research was supported by ONR under MURI grant N00014-08-1-1015 and the NSF under grants 0724806, 0713046, 0847968, and an NSF Graduate Fellowship.

6. REFERENCES

[1] A. Amirbekyan and V. Estivill-Castro. A new efficient privacy-preserving scalar product protocol. In *AusDM*, 2007.

[2] M. J. Atallah, F. Kerschbaum, and W. Du. Secure and private sequence comparisons. In *WPES*. ACM, 2003.

[3] M. J. Atallah and J. Li. Secure outsourcing of sequence comparisons. *Int. J. Inf. Secur.*, 4(4):277–287, 2005.

[4] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x?: Anonymized social networks, hidden patterns, and structural steganography. In *WWW*. ACM, 2007.

[5] F. Bancilhon and N. Spyratos. Protection of information in relational data bases. In *VLDB*, pages 494–500, 1977.

[6] 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*, 3(6), 2005.

[7] 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*, 33, 2005.

[8] A. Deutsch and Y. Papakonstantinou. Privacy in database publishing. In T. Eiter and L. Libkin, editors, *ICDT*, volume 3363 of *LNCS*, pages 230–245. Springer, 2005.

[9] R. Dorfman. The detection of defective members of large populations. *Ann. Math. Statist.*, 14:436–440, 1943.

[10] D.-Z. Du and F. K. Hwang. *Combinatorial Group Testing and Its Applications, 2nd ed.* World Scientific, 2000.

[11] W. Du and M. J. Atallah. Protocols for secure remote database access with approximate matching. In *E-Commerce Security and Privacy: Adv. in Info Security*, volume 2, pages 87–112. 2001.

[12] W. Du and M. J. Atallah. Secure multi-party computation problems and their applications: A review and open problems. In *NSPW*, 2001.

[13] C. L. DuBois. UCI network data repository, 2008.

[14] C. Dwork, F. McSherry, and K. Talwar. The price of privacy and the limits of LP decoding. In *STOC*, pages 85–94. ACM, 2007.

[15] D. Eppstein, M. T. Goodrich, and D. S. Hirschberg. Improved combinatorial group testing for real-world problem sizes. In *WADS*, Lecture Notes Comput. Sci. Springer, 2005.

[16] M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Adv. in Cryptology — EUROCRYPT*, 2004.

[17] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. A walk in facebook: Uniform sampling of users in online social networks. *CoRR*, abs/0906.0060, 2009.

[18] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC*, pages 218–229. ACM, 1987.

[19] M. T. Goodrich. The mastermind attack on genomic data. In *IEEE Symposium on Security and Privacy*. IEEE Press, 2009.

[20] R. Gross, A. Acquisti, and H. J. Heinz, III. Information revelation and privacy in online social networks. In *WPES*. ACM, 2005.

[21] 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*, 64(2):161–166, 1992.

[22] C. Jernigan and B. Mistree. Gaydar: Facebook friendships expose sexual orientation. *First Monday [online]*, 14(10), 2009.

[23] S. Jha, L. Kruger, and V. Shmatikov. Towards practical privacy for genomic computation. In *IEEE Symp. on Security and Privacy*, pages 216–230, 2008.

[24] W. Jiang, M. Murugesan, C. Clifton, and L. Si. Similar document detection with limited information disclosure. In *ICDE*, pages 735–743. IEEE, 2008.

[25] M. Kantarcioğlu, J. Jin, and C. Clifton. When do data mining results violate privacy? In *KDD*, pages 599–604. ACM, 2004.

[26] D. Knuth. *The art of computer programming*. Addison-Wesley, 1973.

[27] 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*, 30(4):330–342, 2008.

[28] G. Miklau and D. Suciu. A formal analysis of information disclosure in data exchange. *J of Comp and Sys Sciences*, 73(3):507–534, 2007.

[29] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE SP*, pages 111–125, 2008.

[30] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *IEEE SP*, pages 173–187, 2009.

[31] B. Pakendorf and M. Stoneking. Mitochondrial DNA and human evolution. *Annual Rev. Genomics Hum. Genet.*, 6:165–183, 2005.

[32] 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*, 35:D823–D828, 2007.

[33] M. Ruszinkó. On the upper bound of the size of the r-cover-free families. *J. Combin. Th. Ser. A*, 66:302–310, 1994.

[34] Y. Sang and H. Shen. Privacy preserving set intersection based on bilinear groups. In *ACSC*, pages 47–54, 2008.

[35] L. A. Stern and K. Taylor. Social networking on Facebook. *J of the Comm., Speech & Theatre Association of ND*, 20:9–20, 2007.

[36] A. Traud, E. Kelsic, P. Mucha, and M. Porter. Community structure in online collegiate social networks. arXiv:0809.0960, 2008.

[37] J. R. Troncoso-Pastoriza, S. Katzenbeisser, and M. Celik. Privacy preserving error resilient DNA searching through oblivious automata. In *ACM CCS*, pages 519–528, 2007.

[38] J. Vaidya and C. Clifton. Secure set intersection cardinality with application to association rule mining. *JSC*, 13(4):593–622, 2005.

[39] A. C. Yao. Protocols for secure computations. In *FOCS*, pages 160–164, 1982.