Computational Complexity and Information Asymmetry in Election Audits with Low-Entropy Randomness

Nadia Heninger
Princeton University
nadiah@cs.princeton.edu

Abstract

We investigate the security of an election audit using a table of random numbers prepared in advance. We show how this scenario can be modeled using tools from combinatorial graph theory and computational complexity theory, and obtain the following results: (1) A randomly generated table can be used to produce a statistically good election audit that requires less randomness to be generated in real time by the auditors. (2) It is likely to be computationally infeasible for an adversary to compute, given a pre-prepared table of random numbers, how to minimize their chances of detection in an audit. (3) It is computationally infeasible to distinguish a truly random table from a malicious table that has been modified to decrease the probability of detecting cheating in certain precincts.

1 Introduction

In this paper, we answer open problems posed by Rescorla [15] concerning adversarial attacks on auditing schemes using randomness tables. We show that truly random tables can enable successful audits, and that it is computationally infeasible to optimize attacks that attempt to use the joint distribution of precincts in such tables.

To answer these problems, we introduce an analytic model for table-based auditing that draws on combinatorial graph theory. Using our analytic model, we also show that a book of random numbers generated by an adversary (rather than a trusted third party) can make possible difficult-to-detect attacks. Our results are related to those in a recent paper of Arora et al. [2] on the hardness of detecting certain kinds of malfeasance in financial derivatives.

Background In a post-election vote tabulation audit, a subset of precincts, ballots, or other audit units are selected for a manual recount, and the results are compared to the preliminary election results in order to gain some measure of confidence in the validity of the election. In the following, we will generally discuss selecting precincts to audit, but from the theoretical perspective of this paper, the results apply equally to ballots, machines, or any other sampled element.

In order for the audit to be informative, the set of precincts should be statistically representative of the election as a whole, and difficult to predict beforehand. A simple, statistically valid method of generating such a sample is to select audited precincts at random from the set of all precincts, either uniformly or with probability weighted by size or other features that need to be normalized over the population.

However, in the context of an election audit, there are two additional desirable properties: first, that the procedure used to generate the set of precincts should be *observable*, so that the population at large can be sure that the election audit (and therefore the election) is actually valid; and second, that the procedure should be *efficient* in terms of the human effort required to perform the audit.

To address these goals, a variety of procedures have been proposed to select precincts for auditing. One is to use a pseudorandom number generator seeded by a dice roll [5] or by stock market fluctuations [8]. Since a cryptographically secure PRNG or randomness extractor is an involved algorithm and likely requires a computer to execute, it has been argued that these methods are not sufficiently observable by an average citizen who does not understand or trust computers. Even an expert may find it difficult to verify that hardware or software implementations work as required.

Several algorithmically simpler methods to produce randomness have been proposed, including rolling dice [6, 10, 13] and non-cryptographically secure procedures that can be executed on a pocket calculator [16].

The difficulty with these physical methods of generating randomness is that the randomness is expensive, particularly if one demands a high-quality source of randomness. Dice rolling seems to be an ideal source of randomness if only a few bits of entropy are required, and thus methods have been proposed to expand the amount of randomness produced. One suggested solution is to use a pre-prepared table of random numbers such as "A Million Random Digits" [7] as an intermediary to expand the random digits [6]. One might roll some dice to generate a starting position in the table, and then sequentially read numbers starting from that position until a set of precincts to be audited has been completed.

Randomness tables and their limits Using a pre-published table like this can be seen as revealing some information about the audit before the election has even occurred. Such a random table seems to be *observable*, in the sense from above that a citizen should be able to examine the table for problems, but the hesitant citizen might protest that this table by its construction cannot contain as much randomness as a true random sample.

Rescorla [15] suggested that an adversary planning to manipulate an election might use the published entropy source to minimize their chances of detection. He introduces the idea of an *adversarial attack* on an election audit with published tables of randomness, and uses simulations to show that an adversary can use this information to decrease the chances of detection in an audit.

There are several methods an attacker might use to minimize their chances of detection. The one explored by Rescorla is to exploit the normal variance in the occurrences of each digit in a table, so that the attacker chooses a set of precincts to attack from those that occur the least often. Using the individual distribution of elements, his simulations show that an attacker can reduce the chance of detection by as much as 10%, and force an auditor to audit as many as 50% more precincts to gain the same level of confidence.

He left the following problems open: Is there an analytic model for such auditing schemes? And is it possible for an attacker to exploit the joint distribution of elements in the table to further decrease chances of detection?

Our results We introduce an analytic model for randomness amplification schemes of this type. Using our analytic model, we are able to answer the open problems posed by Rescorla and obtain the following results:

- *Truly random tables* A reasonably-sized table of random numbers that is truly generated at random can with high probability result in an audit that will detect fraud with any desired confidence level, while requiring the auditors to generate fewer random bits on the fly.

  However, in order to gain the same level of confidence, such a scheme requires auditing more precincts than would be necessary had the sample been generated truly at random.

- *Adversarial attacks on tables* An adversary cannot in general efficiently use a pre-published table of random numbers to decrease chance of detection in fraud they plan to commit beyond a particular range of values.

In addition, our analytic model allows us to draw a connection to another recent application of graph theory that gained widespread attention: the work of Arora et al. [2] on the computational hardness of detecting a financial derivative that has been adversarially constructed to fail.

Using this connection, we are able to pose and answer an additional question suggested by Rescorla's work: Can an adversary construct a table that makes efficient attacks possible? This is a different model from the one adopted by Rescorla, which assumed that the randomness table was honestly chosen, but known to an adversary who would use it to choose precincts to attack. In this variant of the attack, the adversary is more powerful, as he first creates the randomness table used in the audit. The table is specially constructed so that a small number of precincts are less likely to be audited and can be corrupted at will. Because the table he creates is public, however, it can itself be subject to audits and must not be distinguishable from random.

Drawing on the connection to [2], we show that a powerful and undetectable attack is possible in this model:

- *Adversarially-generated tables* An adversary who has control over the generation of the table of random numbers but not the roll of the dice used to choose an offset can create a malicious "random" table that is computationally indistinguishable from a truly random table which decreases the chance of detecting fraud in a pre-selected set of precincts.

We give an example showing how using this sort of manipulation, an adversary can reduce the probability of detection of an attack from 60% to 2%.

Our methods We analyze the problem by understanding it as a question about combinatorial graphs. In particular, it turns out that a good random table will correspond to a graph with good expansion properties. Such graphs are often used in theoretical computer science to reduce the amount of randomness required by a randomized algorithm, and the study of such graphs is an active field of research which is interesting in its own right. This new look at the problem of analyzing random tables allows us to apply tools from the rich theory of expander graphs.

2 Problem statement and mathematical models

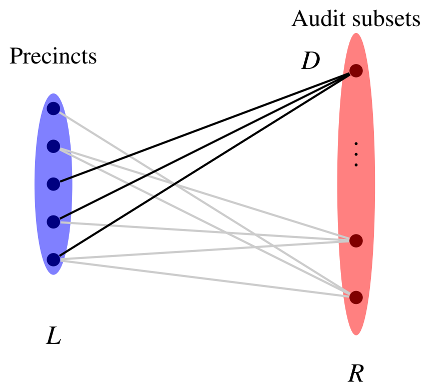

Figure 1: Modeling a table of random numbers using a bipartite graph. We create one node for each precinct, and one node for each subset of precincts in the table, and draw an edge between an auditing subset and a precinct if the precinct is contained in the auditing subset.

The auditing game Rescorla introduces an "auditing game" to model the interaction between an attacker who wishes to cheat in an election without detection, and an auditor who designs an auditing procedure to detect election abnormalities.

In this game, the attacker selects a subset of precincts to modify before the election. (For simplicity, we will talk about precincts, but they could be voting machines or individual ballots.) The election takes place, and the auditor selects a set of precincts to audit. If the auditor selects a precinct that the attacker has modified, then the attack is detected and the attacker loses the game. In reality, the fraud could remain undetected through other means, but this provides a conservative model for the attacker.

Given a PRNG, a table of random numbers, or other information about the auditing procedure, the adversary's goal is to use this information to minimize the chance of detection while cheating in as many precincts as possible.

Random number tables The particular scenario we investigate is that of using a pre-published table of random numbers to reduce the amount of randomness required to select a sample of precincts to audit. In the most natural procedure, dice rolls or another procedure is used to choose an offset in the table uniformly at random, and then a sequence of numbers is read from the table. The sequence of numbers will determine the set of precincts to audit.

If the number of precincts to be audited is D, the procedure above is equivalent to publishing the list of D numbers following each offset in the table.

We will model this procedure using a bipartite graph. See Figure 1 for an illustration.

On the left side, we have L vertices, one for each precinct. On the right side, we have R vertices, one for each audit subset in the table, corresponding to each offset. There will be an edge between vertex l and vertex r if the precinct l is in the audit set corresponding to the offset of r. We will denote by $\Gamma(S)$ the number of neighbors of a set S of vertices in the graph. The degree of each vertex r in R is D, since every audit set contains D precincts, and the degree of each vertex of L is the number of audit sets it appears in.

If an offset of the table is chosen uniformly at random, then the probability that precinct l will be audited is proportional to its degree in the graph described above, $\Pr[l \text{ audited}] = \Gamma(l)/R$. If cheating or other abnormalities occurred in a set S of precincts, the probability in our auditing game that it will be detected is the probability that any of those precincts is audited. This probability is proportional to the number of neighbors of the set,

$$\Pr[S \text{ audited}] = \Gamma(S)/R.$$

See Figure 2 for an illustration.

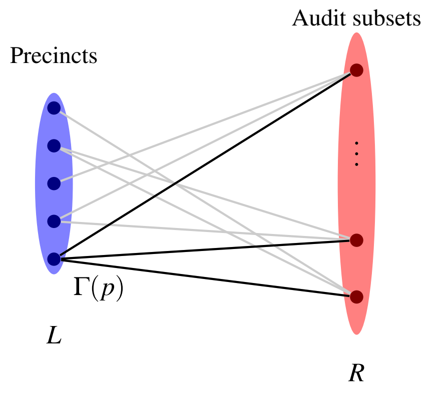

Figure 2: Using the properties of the graph to calculate the statistics of the audit. If a random audit sample is chosen from the nodes on the right, the probability that a particular precinct (or set of precincts) will be audited is proportional to the number of its neighbors on the right.

3 Random tables give good audits

The first observation that this setup allows us to make is that with a properly constructed table of random numbers, the audit procedure described above will give a sample of precincts to audit that requires less randomness than individual dice rolls to select each precinct. Standard statistical methods can be used to give lower bounds on the validity of the resulting audit.

However, there is a trade-off to be made: using a table of random numbers vastly decreases the amount of randomness that needs to be generated on the fly, but in order to get the same degree of confidence in the audit, more precincts need to be audited than if one were selecting precincts to audit truly at random. (Alternatively, one can audit the same number of precincts as before and adjust the confidence of the audit downward.)

We adapt methods used in the analysis of expander graphs to bound the effectiveness of an audit done in this way.

3.1 Expander bounds

To begin with, we will make the simplifying assumption that the dice rolls do not select an arbitrary offset in the table, and instead are used to select an offset at a multiple of D, so that each of the audit sets in the table corresponds to a non-overlapping window. (Note that the resulting book will contain RD entries, and our results can no longer be directly compared to Rescorla's.)

Then the bipartite graph constructed above is a *random bipartite graph*, constructed by selecting D random neighbors on the left for each vertex on the right. It is well-known that such graphs have good *expansion* properties, that is, that all subsets of vertices will have many neighbors. Expander graphs are often used in algorithms in theoretical computer science to reduce the amount of randomness required by a randomized algorithm, among many other applications. See [9] for an excellent overview.

In this case, we are examining unbalanced bipartite graphs, and the effectiveness of the audit is strongly related to the expansion of the precinct vertices.

Definition 1. *A bipartite graph has* left expansion c*, if for all sets of vertices* $S \subseteq L$ *of size* $|S| < \alpha|L|$,

$$\Gamma(S) \geq c|S|.$$

where $\Gamma(S)$*, the size of the neighborhood of* S*, is the number of vertices in* R *with at least one neighbor in* S*, and* $0 \leq \alpha \leq 1$.

Essentially, this definition states that every small subset of vertices on the left-hand side has a large number of neighbors on the right-hand side. For a good expander, the ratio c of neighbors to vertices should be close to the the average degree of the left-hand vertices.

If we have a lower bound on the expansion of the graph representing our table, this allows us to calculate an easy lower bound on the confidence of an audit using that table. If we would like to detect cheating in at least B precincts and we know that our graph has expansion at least c, then any set of B precincts has at least cB neighbors in R, and thus the probability that a randomly selected vertex (audit set) from R has a neighbor among our B (includes one of the B to be audited) is at least cB/R.

It is known that asymptotically, a random graph is likely to be a good expander. Unfortunately, the graph sizes generated by such proofs are too large to be of use for a human-run auditing program.

As a first attempt, we can look at existence proofs for expanders for inspiration. [12, Ch. 5.3], for example, gives an existence proof via the probabilistic method of graphs with good right expansion. We adapt this method for our needs, first to bound the actual probability that a graph is a good expander, and with a bit more thought to bound the left expansion instead of right.

Form the graph by connecting each vertex in R to d random vertices from L, and calculate the probability that all subsets of vertices $S \subseteq L$ of size less than αL (for $0 < \alpha < 1/c$) have expansion greater than c in R. The graph will have probability at least $1-p$ of being an expander if

$$L(R/c)^c e^{-d(R/L-\alpha c)} e^{1+c} < p/(1+p).$$

However, this bound does not give useful numbers for plausible graph sizes for the application of vote auditing. (The problem is that ensuring this property for every very small set introduces a lot of excess in the union bound we're using.)

Instead, we will use a weaker notion of expansion, and ask only that subsets of vertices of a certain fixed size have expansion c. This results in a lower bound on the confidence of an audit of that size. In particular, if we would like to detect cheating that has occurred in at least s precincts, then if all sets of vertices of size s have at least cs neighbors in R, our audit will have confidence at least sc/R of detecting cheating that occurred in at least s precincts.

We can give a generous bound the probability that every such subset s has the required expansion. The following expression calculates a union bound on the probability that no subset s of L has fewer than cs neighbors. Since our edges are coming from R, the probability that all of the s edges are in our desired cs set is the probability that all the other edges from R land in $L-s$.

$$\Pr[\text{good expansion}] \geq 1 - \binom{L}{s}\binom{R}{cs}\left(\frac{L-s}{L}\right)^{d(R-cs)}$$

We will do a brief calculation to illustrate the trade-off between randomness and audit size.

Example 1. *We have an election with 5000 precincts, and wish to guarantee that there were abnormalities in no more than 5% of precincts with confidence 80%.*

In the case of a truly random audit, we should audit at least 32 *precincts to obtain these confidence levels, since*

$$\prod_{i=0}^{31} \frac{5000-250-i}{5000-i} < 0.2.$$

Thus the number of bits of randomness in this case that we need to generate is at least

$$\lg \binom{5000}{32} > 275.$$

Now we compare to the case of a pre-computed table of random numbers. We'll imagine that we are provided with a book that has 200,000d entries, where d will be the size of the resulting audit.

The confidence of the resulting audit will be $250c/200000$*, so we need* $c = 640$*. Setting* $d = 50$*, the probability that a randomly constructed graph from above has our desired c is*

$$1 - \binom{5000}{250}\binom{200000}{640\cdot 250}\left(\frac{5000-250}{5000}\right)^{50\cdot 40000} > 1 - 10^{-600}$$

so we will almost certainly have the confidence we need.

Thus we will need to audit 50 precincts, but the number of bits of randomness needed to generate the audit sample from the book is

$$\lg 200000 < 18.$$

We note that the calculations above make liberal use of the union bound, and thus the bounds should be able to be improved by using more sophisticated methods.

4 Algorithmic attacks

The previous section shows that it is possible to do a good statistical audit with the help of truly random tables. However, this does not eliminate the possibility that an attacker might be able to use the table to optimize an attack within these parameters. Rescorla [15] discusses the case of an attacker who calculates the fraction of times each precinct occurs individually in the audit and attacks those precincts that occur the least often. He poses the question of how one might construct an optimal attack.

Using the framework introduced earlier, we see that the adversary would like to tamper with the set of precincts that has the fewest neighbors in the graph representing the table of audit sets; in other words, to find a relatively small set of vertices in a bipartite graph that has minimum expansion.

This is a natural combinatorial optimization problem that arises in many applications, but until recently little was known about its difficulty. Several recent results suggest that it may be quite difficult to solve in general.

Very recently, Raghavendra and Steurer [14] outlined a connection between the problem of approximating the small-set expansion of a graph and the unique games conjecture, a notorious conjecture in algorithms that would imply strong hardness results for many well-known approximation problems in combinatorial graph theory [11]. In particular, they conjecture that it is NP-hard to distinguish whether the small set expansion of a graph is very large or very small, and show that their conjecture implies the unique games conjecture.

Even more recently, Arora, Barak, and Steurer gave a subexponential time approximation algorithm for the small-set expansion problem [3]. Given a graph on n vertices that contains a set of size k with neighborhood of size ck, in time $\exp(n^{O(c^{1-\beta})})\,\mathrm{poly}(n)$, they can find a set of k vertices with neighborhood of size at most $kc^{\beta/3}$. The graphs we use in our examples here have thousands of vertices; thus the stated running time here will only be feasible for c much smaller than 1; that is, if the graph contains a shrinking set. Detecting a subset with small but nonshrinking expansion appears to remain infeasible.

Additionally, the results above are stated for arbitrary graphs, and our graphs are bipartite. There may be a solution in some special cases, but as of yet there is no indication that bipartite graphs should be much easier in general.

5 Malicious auditing

The difficulty that an adversary has in optimizing an attack strategy is not specific to an attacker. Observe that the desired outcome of the attacker, to find a subset of precincts that is unlikely to be audited, is very closely related to the desired outcome of an auditor, who wishes to make sure that all subsets of precincts are equally likely to be audited. If the auditor could calculate subsets of precincts that were unlikely to be audited, he would be well on his way to his goal. In fact, it is difficult for *anyone* to efficiently verify that the random table has the randomness properties that are desired.

This means that it is possible to construct a malicious table of "random" numbers that cannot be efficiently distinguished from a good, truly random table. Such a table might be constructed to decrease the probability that an attack on the election was detected in an audit. In a more practical sense, it means that there is no efficiently computable way to verify that a table of random numbers is truly random, or that it has the properties we outlined in Section 3.

If we are sure that a table was truly generated at random, then we can as above calculate the probability that such a table has the properties we want, but if we are handed a table that was generated opaquely, it is not possible to verify that it was honestly generated.

This suggests the following attack on our table-aided vote auditing scheme: an attacker who has control over the table of random numbers (and knows the assignment of table entries to precincts) but not the roll of dice used to later select an offset within the table can create a malicious table that will decrease the probability of detecting fraud within a pre-specified set of precincts.

The technique we will use to generate a malicious table is to place a *planted dense subgraph* in the bipartite graph representing the table. Within certain parameters, it is believed to be computationally difficult to detect such a graph, and currently there is no good algorithm for doing so. The hardness of this *k-densest subgraph problem* has been proposed as the basis of a cryptosystem by Applebaum et al. [1], and used by Arora et al. [2] to show that it is possible to construct financial derivatives that are indistinguishable from fair derivatives, but that are designed to cheat. Currently the best known algorithm for solving the k-densest subgraph problem is a polynomial-time algorithm giving an $O(n^{1/4})$ approximation due to Bhaskara et al. [4].

The problem of computing the densest subgraph is related to the problem of computing the small-set expansion of a graph [14]. A dense subgraph limits the expansion of the vertices in the dense subgraph, and a graph with good expansion cannot have a very dense subgraph.

In contrast to the attacks outlined by Rescorla, for which an auditor can try to compensate by decreasing the estimated confidence of an audit to correspond to the advantage an attacker might get (in the manner of Section 3), this kind of attack is impossible to predict or compensate for without the secret knowledge of the hidden dense subgraph.

Arora et al. formalize an asymptotic intractability assumption on the hardness of detecting a planted dense subgraph in a random graph. Since we are interested in exploring the ramifications of this assumption for some concrete families of parameters, we will boldly adapt a non-asymptotic version of the assumption. For our version of the assumption, we have both eliminated asymptotic factors in the ratios between the vertex degree inside and outside of the planted subgraph, as well as apply the assumption to fixed large graph sizes.

Should an efficient algorithm be found for certain ranges of values or constants, the precise formulation of the assumption may need to be changed.

Densest subgraph assumption For large graph sizes, it is computationally intractable to distinguish between a graph drawn from distribution

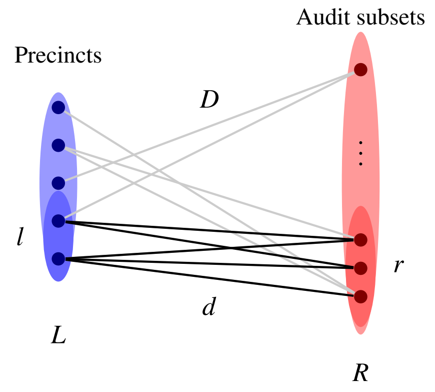

Figure 3: Manipulating the audit by planting a dense subset of edges in the graph. A small subset of the audit sets are highly connected with a selected subset of precincts; this means that these precincts appear less often in the remaining audit sets and are less likely to be audited overall.

- $\mathscr{R}$ obtained by choosing for every right vertex D neighbors uniformly at random on the left.

- $\mathscr{P}$ obtained by choosing r vertices from the right and l vertices on the left, choosing d neighbors for each of the r vertices at random from the l vertices, and $D-d$ neighbors at random from the remaining $R-r$ vertices, and apportioning the edges for vertices in the $R-r$ vertices remaining so that the degree distribution remains constant.

for the parameters $d \approx \sqrt{D}$ and $r \approx l\sqrt{R/L}$.

In the following analysis, we will compare the attacker's advantage to the likelihood of detection in a truly random election, instead of the adversarial case we outlined in Section 3. This much simplifies the analysis. It is also sensible, since the bounds we derived in Section 3 are a different phenomenon from the type of manipulation we are exploring here. (In any case, Section 4 shows that an adversary will have difficulty taking advantage of graph properties they do not control.)

Recall that the auditing process corresponds to picking one subset of precincts and auditing that set. The particular subset corresponds to the left endpoints of the edges originating from one vertex on the right. The adversary chooses a set of l vertices from the left-hand side to modify.

In the case of the truly random graph in $\mathscr{R}$, we can calculate the expected number of edges from the r vertices on the right to the l vertices on the leftt:

$$\#\text{ edges } r \rightarrow l : Drl/L.$$

as well as the expected number of edges from the $R-r$ remaining vertices to the l vertices on the left.

$$\#\text{ edges } (R-r) \to l : D(R-r)l/L.$$

Thus the total number of edges into l is DRl/L.

The adversary wins if his corrupted set of l vertices has no overlap with the audited set, thus the goal of the planted dense subgraph is to increase the number of edges from l to a small subset of vertices r, thus decreasing the total number of audit sets with an overlap with l.

In the case of the planted graph from $\mathscr{P}$, the number of edges will be:

$$\begin{aligned}\#\text{ edges } r \to l : &\, dr + (D-d)rl/L \\ &= Drl/L + dr(L-l)/L\end{aligned}$$

$$\begin{aligned}\#\text{ edges } (R-r) \to l : &\, DRl/L - (dr + (D-d)rl/L) \\ &= D(R-r)l/L - dr(L-l)/L\end{aligned}$$

where the second follows because we construct the graph to preserve the same average degree for all vertices.

Because the vertex on the right (and thus the audited set) is chosen uniformly at random, the probability that the adversary will succeed is precisely the probability that a random vertex on the right has no neighbors in l.

To simplify the following analysis, we will model the probability of detection as a balls in bins process and use the Poisson approximation. Thus if we throw m balls into n bins, the probability that a particular bin has at least 1 ball is $1-e^{-m/n}$.

In this case, we are throwing edges from l into R and asking what the probability that at least one of the edges has an endpoint in a randomly selected vertex in R.

For the case of the truly random graph from $\mathscr{R}$, we have DRl/L edges into R vertex bins, so

$$\Pr[\text{detection}] = 1 - e^{-Dl/L}.$$

For the biased case, the probability of detection will be the probability that we select a vertex from r times the probability that the attack is detected in that case, and similarly for $R-r$:

$$\begin{aligned}\Pr[\text{detection}] &= \tfrac{r}{R}\left(1 - e^{-(Drl/L + dr(L-l)/L)/r}\right) \\ &\quad + \tfrac{R-r}{R}\left(1 - e^{-(D(R-r)l/L - dr(L-l)/L)/(R-r)}\right) \\ &= 1 - \tfrac{r}{R}\left(e^{-(Dl/L + d(L-l)/L)}\right) \\ &\quad - \tfrac{R-r}{R}\left(e^{-(Dl/L - dr(L-l)/(L(R-r)))}\right)\end{aligned}$$

It is not clear that this kind of biasing would always benefit an attacker, but in fact the following lemma shows that it always will.

Lemma 1. *For all $x > 0$ and all b,*
$1 - e^{-x} \geq 1 - \left(\frac{r}{R}e^{-(x+b)} + \frac{R-r}{R}e^{-(x-br/(R-r))}\right)$.

In particular, the lemma holds for $x = Dl/L$ and $b = dr(L-l)/L$.

Proof. Apply the arithmetic geometric mean inequality and solve.

$$\begin{aligned}&\frac{r}{R}e^{-(x+b)} + \frac{R-r}{R}e^{-(x-br/(R-r))} \\ &\quad\geq \sqrt[R]{e^{-(x+b)r}e^{-(x-br/(R-r))(R-r)}} \\ &\quad= \sqrt[R]{e^{-rx-br}e^{-(R-r)x+br}} \\ &\quad= e^{-x}\end{aligned}$$

There will be equality when $r = R-r$. □

Thus it is in the attacker's advantage to make the difference between the two terms as large as possible.

We will use our example from earlier to calculate the difference that this kind of cheating can make.

Example 2. *We have $L = 5000$, $l = 250$, and $D = 32$.*

In the random case, our probability of detecting cheating (on average this time, and not adversarial) is about

$$1 - e^{-Dl/L} \approx 0.798.$$

To construct an altered graph, we have $R = 200{,}000$ and $r = l\sqrt{R/L} = 1581$. We choose $d = 5 < \sqrt{32}$. Then $1 - \frac{r}{R}\left(e^{-(Dl/L+d(L-l)/L)}\right) - \frac{R-r}{R}\left(e^{-(Dl/L-dr(L-l)/(L(R-r)))}\right) \approx 0.792$.

Thus the adversary has lowered his chances of detection by nearly 1% in a way that is completely undetectable to any auditor.

Conceptually, this is an important case, as it shows that the effect can be real even with realistic parameters. However, the benefit that the adversary derives increases very strongly as the size of the table increases. A ballot-based audit might potentially involve a larger necessary table size. To demonstrate, we will take an election with 100 million voters, potential for 2% fraud, and a book with 2 million entries.

Example 3. *We have $L = 100 \times 10^6$, $l = 2 \times 10^6$, and $D = 50$.*

In the random case, our probability of detecting cheating is about

$$1 - e^{-Dl/L} \approx 0.632.$$

Now let $R = 2 \times 10^6$ and $r = l\sqrt{R/L} = 282843$. We choose $d = 7 < \sqrt{50}$.

Then $1 - \frac{r}{R}\left(e^{-(Dl/L+d(L-l)/L)}\right)$
$- \frac{R-r}{R}\left(e^{-(Dl/L-dr(L-l)/(L(R-r)))}\right) \approx 0.022.$

That is, the adversary has lowered the chance of detection from above 60% *to about* 2%.

It is worth noting that the size of the table is small enough that it will not be likely to have the good expansion properties from Section 3.

If the size of the audit book R is increased by a factor of 10, the attack is largely mitigated, and the probability of detection falls from 60% *to only about* 50%. *Conceptually, increasing or decreasing the size of the book in the non-adversarial model has no effect on the probability of detecting randomly located fraud in the audit. It only affects the adversarial analysis.*

This final example is so striking that it seems to call into question the assumption that detecting such fraud is computationally intractable. Nevertheless, it does not seem to open any avenues for a more efficient algorithm to solve the densest k-subgraph problem.

6 Practical considerations and mitigations

Since the use of a random number table in a post-election audit is still a hypothetical, there are no existing procedures respecting such a table that we can use to evaluate the practical difficulty of carrying out the attacks outlined above.

The simplest mitigation against the attack outlined in Section 5 would be to use a table that predates the idea of using such tables to audit elections. If a special-purpose table is to be generated, however, it should not be difficult for an attacker to generate suitable entries for a malicious table. From that point on, the difficulty of supplying such a table to an unwary government should be no greater than, for example, convincing a government to use black box electronic voting machines.

We will discuss several potential mitigations.

Observable randomness generation For such a table to be trustworthy, the process used to generate it needs to be trusted and carefully observed throughout the generation of the table. For a table with millions of entries, this could be a potentially onerous requirement, and one that is difficult for the population of voters to verify for themselves. The introduction to [7] provides an illustration of the difficulty of generating such numbers through physical means in 1947.

We note that the procedures used to extract randomness from physical sources now (for example, to generate uniform randomness from the fluctuations of the stock market or from a quantum oscillator) generally run inputs of imperfect randomness through a randomness extractor, a numerical algorithm designed to transform a biased source of randomness into a truly uniform output. However, a numerical algorithm that has the desired properties is a pseudorandom generator, so if the electorate wishes to avoid having to trust an algorithmic pseudorandom number generator, we've merely complicated the story.

Multiple sources Alternatively, one could let several different parties generate tables, and combine them together by, for example, xoring the entries. As long as each party commits to his entry before seeing any of the others at least one of the tables was truly random, the output of this process remains truly random.

Verifying tables before use Once such a table is in use, all of its entries should be published in multiple locations, along with details of the selection procedure whenever the table is used, to prevent substitution of a malicious table after generation.

Assigning random IDs to precincts One proposed strategy to avoid the attack outlined in Section 5 is to choose a separate assignment of precincts to table entries. Unfortunately, more randomness is required to choose the assignment of precincts than would be required to choose a completely random subset of precincts to audit.

7 Lessons

The attack outlined in Section 5 is almost certainly implausible in a real election. However, it serves to illustrate the lesson that when assessing the statistical validity of an audit, it is vitally important to understand all of the inputs to the process.

Randomness is a powerful tool, but its power relies on the randomness being properly generated. Once a sequence of random numbers has been generated, it may be computationally difficult to verify that the sequence has all the properties that a truly random sequence should hold.

In essence, using a table of random numbers in the way we study in this paper is equivalent to substituting the table for an algorithmically generated PRNG or randomness extractor. In fact, we can use similar tools to analyze both cases.

The fact that a table of random numbers seems more "accessible" than a baroque algorithm for generating pseudorandom numbers does not mean it is any more trustworthy; on the contrary, it is much easier to have statistical confidence in the analysis of an algorithm that has been designed to be easy to analyze than it is to be confident that a table of numbers has been properly generated, no matter how many statistical tests we run on the numbers.

8 Conclusion

The analysis provided in this work allows us to distinguish several security properties for the use of random tables in election audits.

On a positive note, the expansion properties of random graphs can be used to give absolute security that the table does not contain harmful statistical abnormalities. We show how random tables can be used in a statistically sound way to reduce the amount of randomness required to perform an audit, if one is willing to accept a tradeoff in either confidence levels or the number of precincts that must be audited.

On the other hand, the hardness of optimizing an attack against the audit gives computational security. Even if a table does not satisfy the absolute security properties above, an attacker is unlikely to find a bad set to exploit that an auditor cannot also find. Since the statistics of the random sample are dependent only on the proportion of the sample and on the size of the book, this may allow the use of a table that is smaller than the strict standards of absolute security might require, since it would be computationally infeasible to find a weakness.

And finally, the adversarially-designed table gives complete insecurity. A vulnerability can be built into such a table that is undetectable for anyone who does not know the secret.

Beware of statisticians bearing gifts.

Acknowledgements

I am grateful to Henry Cohn for help with the calculations, to Alexandra Kolla and David Steurer for references on eigenvalues and expansion, and to Hovav Shacham, Eric Rescorla, Alex Halderman, and the anonymous reviewers for helpful suggestions and clarifications. This material is based upon work supported under a National Science Foundation Graduate Research Fellowship.

References

[1] Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In *STOC '10: Proceedings of the 42nd ACM symposium on Theory of computing*, pages 171–180, New York, NY, USA, 2010. ACM.

[2] Sanjeev Arora, Boaz Barak, Markus Brunnermeier, and Rong Ge. Computational complexity and information asymmetry in financial products (extended abstract). *Innovations in Computer Science*, January 2010.

[3] Sanjeev Arora, Boaz Barak, and David Steurer. Subexponential algorithms for unique games and related problems. manuscript, 2010.

[4] Aditya Bhaskara, Moses Charikar, Eden Chlamtac, Uriel Feige, and Aravindan Vijayaraghavan. Detecting high log-densities: an $O(n^{\frac{1}{4}})$ approximation for densest k-subgraph. In *STOC '10: Proceedings of the 42nd ACM symposium on Theory of computing*, pages 201–210, New York, NY, USA, 2010. ACM.

[5] Joseph A. Calandrino, J. Alex Halderman, and Edward W. Felten. In defense of pseudorandom sample selection. *USENIX/ACCURATE Electronic Voting Technology Workshop 2008*, July 2008.

[6] Arel Cordero, David Wagner, and David Dill. The role of dice in election audits—extended abstract. *IAVoSS Workshop on Trustworthy Elections 2006 (WOTE 2006)*, June 2006.

[7] RAND Corporation. *A Million Random Digits with 100,000 Normal Deviates*. American Book Publishers, 2002.

[8] Aleks Esseks, Jeremy Clark, Richard T. Carback III, and Stefan Popoveniuc. The punchscan voting system: Vocomp competition submission, 2007.

[9] Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bull. Amer. Math. Soc.*, 43:439–561, 2006.

[10] David Jefferson, Elaine Ginnold, Kathleen Midstokke, Kim Alexander, Philip B. Stark, and Amy Lehmkuhl. Evaluation of audit sampling models and options for strengthening California's manual count. California Secretary of State, July 2007.

[11] Subhash Khot. On the unique games conjecture. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:3, 2005.

[12] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[13] Lawrence Norden, Aaron Burstein, Joseph Lorenzo Hall, and Margaret Chen. Post-election audits: Restoring trust in elections. Brennan Center for Justice at The New York University School of Law and The Samuelson Law, Technology and Public Policy Clinic at the University of California, Berkeley School of Law (Boalt Hall), August 2007.

[14] Prasad Raghavendra and David Steurer. Graph expansion and the unique games conjecture. In *STOC '10: Proceedings of the 42nd ACM symposium on*

Theory of computing, pages 755–764, New York, NY, USA, 2010. ACM.

[15] Eric Rescorla. On the security of election audits with low entropy randomness. *USENIX/ACCURATE/IAVoSS 2009 Electronic Voting Technology Workshop/ Workshop on Trustworthy Elections (EVT/WOTE 2009)*, August 2009.

[16] Ronald L. Rivest. A “sum of square roots” (SSR) pseudorandom sampling method for election audits, 2008.