A "Sum of Square Roots" (SSR) Pseudorandom Sampling Method For Election Audits

Ronald L. Rivest
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139
rivest@mit.edu

April 25, 2008

Abstract

This note proposes a cute little heuristic method of generating a uniformly distributed pseudo-random number between 0 and 1 for each precinct in an election, for use in selecting a sample of precincts for a post-election audit. A function f is described that takes as input a 15-digit "seed" S and a six-digit precinct number i, and produces a pseudo-random output $x_i = f_S(i)$ between 0 and 1. The seed S is obtained by rolling fifteen dice in a public ceremony. For a 5% audit, precinct i will be audited if $x_i \leq 0.05$. This approach also works well for auditing methods, such as NEGEXP, that set different auditing probabilities for different precincts.

We call the proposed method the "SSR" method, as it is based on taking the fractional part of a sum of three square roots. One of the nice features of this method is that it can be performed on the simplest of pocket calculators (assuming it has a square-root button). Thus, local election officials and/or election observers can easily determine and/or verify whether or not each particular precinct should be audited, once the seed S has been determined at headquarters.

The SSR method should be highly unpredictable to an adversary—an adversary who does not know the seed should have no advantage in determining which precincts to corrupt. The SSR method is also highly efficient, since only 15 dice rolls need to be done, instead of thousands. Finally, it is easily verifiable on a simple calculator.

Keywords: pseudo-random number, pseudo-random function, dice, sampling, post-election audit, square-root, sum of square roots.

1 Introduction

The auditing of elections has become an area of active research and discussion. See [1, 6, 5, 7, 2], for example.

A post-election audit consists of five steps. The first step is to collect all of the initial election results from each precinct; this includes the number of votes for each candidate in each race, which also determines the apparent margin of victory. The second step is to use these collected results to determine the parameters of the audit, such as the sample size or probability that each precinct is to be audited. The third step is to actually select the precincts to be audited; this is done in a randomized manner, typically involving dice. The fourth step is to recount by hand the paper ballots in the selected precincts. Finally, if significant discrepancies are found, the audit may be escalated (to a larger sample). This paper focuses on the third step, the actual selection of the precincts to be audited.

[0]The latest version of this paper is available via: http://people.csail.mit.edu/rivest/publications.html

For an audit to be effective, the set of precincts to be audited must be unpredictable to an adversary. For the audit results to be credible, the selection of the sample must be transparent to the public. Finally, it must be possible to select the sample in a simple manner that is easily accomplished by election officials.

There is precious little literature on the problem of selecting a sample of precincts to audit in a manner that is unpredictable, transparent, and simple.

Cordero et al. [3] give an excellent introduction to the problem, and explain how to use ten-sided dice to pick precincts to be audited. It works well, and can be recommended, when the selection problem to be solved is to pick k objects out of n, *uniformly at random*. That is, each of the n objects is equally likely to be selected. (The objects to be selected could be precincts, or perhaps even ballots, depending on the auditing context.)

However, there is no reason to restrict auditing methods to those that picks precincts with equal probability. For example, a recent paper by Aslam, Popa, and Rivest [1] explores sampling methods that proceed in a rather different manner. These are the "basic" sampling methods, which includes the NEGEXP proposal that is particularly flexibile and efficient.

With a *basic* sampling method, each precinct P_i (for $i = 1, 2, \ldots, n$) has an associated probability p_i that it will be audited. These probabilities may all be equal (e.g., $p = 0.05$ for a "5% audit"), or they may vary. For example, they may depend on the size of the precinct. How these probabilities p_i are determined need not concern us here; the reader is referred to [1] for details relating to the NEGEXP method as an example of this approach.

We can then imagine that for each precinct P_i, we have an associated coin that has probability p_i of coming up "heads" and probability $1 - p_i$ of coming up "tails." We independently flip the coin associated with each of the n precincts; if the i-th coin comes up "heads" then the i-th precinct P_i is audited; otherwise it is not audited.

The problem addressed in this note is how to simulate all of these n biased coin flips in a manner that is unpredictable, transparent, simple, and verifiable. That is, we focus on the implementation of basic auditing methods. For the purpose of this paper, it is not significant which basic auditing method is being used; all that matters is that each precinct have an associated audit probability p_i. (These could all be equal.)

2 Dice-rolling

Suppose we have a state with 5,000 precincts, each with an associated audit probability p_i. How might we arrange the necessary $5,000$ biased coin flips? This section examines a straightforward approach using dice.

2.1 The naive approach—massive dice-rolling

In the naive approach, we would roll say 6 ten-sided dice for each precinct P_i, creating a decimal number $x_i = 0.d_1d_2d_3d_4d_5d_6$ (with d_j the digit from the j-th dice roll). Clearly x_i is uniformly distributed between 0 and 1 (up to the obvious limitation that we only have six digits of precision in x_i; more dice could be rolled for greater precision). If $x_i \leq p_i$, the i-th precinct is audited. Thus, precinct P_i is audited with probability (very nearly) equal to p_i, as desired.

This approach would require $30,000$ dice rolls, which seems excessive, particularly when one adds the requirement that the dice rolls be performed publicly and videotaped for transparency. At ten seconds per roll, this would take over 83 hours. (Even so, this is presumably considerably less work than doing the audit itself.)

2.2 A distributed dice-rolling approach

It seems natural to consider then a distributed approach, where six dice are rolled in each precinct, to determine whether that precinct should be audited. This distributes the effort of the dice-rolling.

However, when the requirements for transparency are added (public viewing and videotaping of the

dice-rolling ceremony), the distributed approach no longer seems very workable.

In particular, one might reasonably be concerned that the necessary security and transparency might be difficult to obtain with such a distributed approach. Election officials in a precinct might be tempted to "do-over" the dice-rolls until a desired outcome (e.g. "no audit") is obtained.

3 Pseudo-random numbers

We therefore now consider a different approach, which greatly reduces the number of (true) dice rolls needed. There are a small number of true dice rolls, that determine a large number of "pseudo-random numbers" through the use of a pseudo-random function.

In this method, a few ten-sided dice (15 in our proposal) are rolled in a central location at a public ceremony, in a public, videotaped, and transparent manner. The results form a master public "seed" number S. In our proposal, they are actually divided into three five-digit seed numbers S_1, S_2, S_3 for our computation.

Then, the election officials of each precinct P_i determine the precinct's "random number" x_i (where $0 \leq x_i \leq 1$), by combining the master seed number(s) S with the precinct number i according to some fixed, public, deterministic function f:

$$x_i = f_S(i) \ . \tag{1}$$

The function f should intuitively be a "pseudo-random function"—the output x_i for an input i should be unpredictable and random-looking. (We'll talk more about properties of f shortly; for our application the requirements are simpler than for the cryptographer's definition of a "pseudo-random function.") We suggest that it is also important that the function f be simple to compute; the purpose of this paper is to suggest such a function f.

The idea of using a pseudo-random function f in this manner is not new; it is described for example in Calandrino et al. [2, Section 3.2], who discuss a number of variations and considerations, such as the idea of using the ballot, instead of the precinct, as the auditing unit. But they do not suggest a particular pseudo-random function to use.

A major benefit of the use of a pseudo-random function is the fact that its outputs are verifiable. The numbers x_i are publicly computable by anyone, since the master seed S is public, the precinct numbers i are public, and the function f is public.

We also assume that the the precinct audit probabilities p_i are also publicly available (or publicly computable, given the election data (precinct sizes, margin of victory, etc.)). See [1] for details of computing p_i for the NEGEXP method. If a "flat audit" is being performed, each p_i is equal to some fixed flat audit rate p (e.g. $p = 0.05$).

Thus, it is straightforward, once the seed S has been published, for anyone to determine whether a given precinct P_i should be audited. In particular, the election officials at precinct P_i can easily do so, as can any citizen with a simple calculator.

First, the probability p_i that precinct P_i is to be audited is determined, given the preliminary election results.

Second, the value of the seed S is obtained.

Then the value x_i of the pseudo-random function f on inputs S and i is determined (see equation (1)). Then, if $x_i \leq p_i$, precinct P_i is scheduled to be audited.

This approach is efficient and transparent. Only a few dice rolls are needed; these are to compute the seed S. Instead of rolling 30,000 dice, election officials need to roll only 15—a reduction by a factor of 2000. This is particularly significant since each dice roll needs to be witnessed and perhaps videotaped.

Once S is determined, election officials can also publish a list of the x_i and p_i values for each precinct. Those precincts with $x_i \leq p_i$ are those that will be audited.

We don't further discuss the (nontrivial) problem of generating the randon number seed S, except to note that it must be done in a public videotaped ceremony that is credibly free of outside influence. Our recommendation would be to base the procedure on the use of dice, although other procedures could be used.

4 Requirements for f

What are the requirements for f for this application?

First, f should be *uniformly distributed*: for each precinct i, random values for S should produce values for $x_i = f_S(i)$ that are uniformly distributed in the interval $[0, 1]$. This is necessary so that the event $x_i \leq p_i$ has probability p_i.

Second, f should be *easy to compute*. We adopt the desideratum that f should be easily computable *using a simple pocket calculator*. We do assume that such a calculator is capable of computing square roots. A calculator producing results with 8 decimal digits of display should suffice.[1] [2]

Third, f should be *unpredictable*. There should be little "structure" relating the values of S and i to values x_i computed. Of course, f is a fixed function, so there is some "structure." But f should appear to behave like a random function.

A cryptographer might call for a pseudo-random function family here, with index S and input i (see Goldreich et al. [4]). However, we have limited ourselves to simple computations on a pocket calculator, so we won't be using the usual cryptographic definitions and complex hash function computations, but rather simpler approximations computable on a pocket calculator. Indeed, our application is also simpler, since in our application the seed is made public. Our security requirement says that knowing f (*but not S or any values $f_S(i)$*) should not give an adversary any advantage in determining which precincts to corrupt.

Fourth, the result should have *sufficient precision*. While we are proposing that it be computable on an 8-digit calculator to an adequate level of precision (perhaps three to five digits), we also would like the result to be computable to greater precision on a more capable calculator. Indeed, we would like the computation to be one that is capable of being extended to arbitrary precision, with calculators or computers capable of working with arbitrary high-precision arithmetic.

[1] We would like the proposed method to work on the simplest of available calculators. We bought a Staples SPL-110 calculator for the amazingly low price of $3.15; it has 8 digits of display, the standard four functions `+`, `-`, `*`, `/`, plus the square-root function, which is all we need.

[2] One could use computers for this task, since the computation of f uses only public inputs and so is verifiable. But elections officials may not have computers readily available, while pocket calculators are very easy to obtain. It is also an interesting challenge to see what can be done with just a pocket calculator.

5 A specific proposal for f

We now propose a particular function f, the "SSR" (Sum of Square Roots) function. We could also call f a "function family," since there is a function f_S for each seed S.

We assume that precincts are numbered $i = 1, 2, \ldots, n$, where n is less than one million (10^6). We shall assume that the precinct number is divided into three blocks i_1, i_2, i_3 of exactly two digits each. For example, for precinct $i = 23951$ we have:

$$i = i_1 i_2 i_3 = \texttt{02 39 51}$$

Note that leading zeros are *not* suppressed; there are exactly three blocks, and each has exactly two digits. (We let $b = 3$ denote the number of blocks, and let $d = 2$ denote the number of digits in each block.)

We assume that the master seed S consists of three blocks S_1, S_2, S_3 of exactly five digits each. Each such number S_j can be obtained by rolling a ten-sided die five times. (The number of blocks is $b = 3$, the same as the number of blocks in the precinct number. The length of each block is exactly $k = 5$ digits. Again, leading zeros are *not* suppressed.)

We emphasize that the determination of the seed numbers is done only once by the central election officials in a public ceremony; the same seed numbers are then used by each precinct.

Continuing our example, we assume that the seed numbers determined are:

$$\begin{aligned} S_1 &= \texttt{27901} \\ S_2 &= \texttt{67556} \\ S_3 &= \texttt{06125} \end{aligned}$$

We now describe how to compute $f_S(i)$, given three precinct index blocks i_1, i_2, i_3 of two digits each and the three seed values S_1, S_2, S_3 of five digits each.

The first step is to form three numbers n_1, n_2, n_3, each of seven digits in length, by concatenating the index blocks with the corresponding seed values. The j-th number n_j is formed by writing the j-th two-digit block i_j of the precinct number followed by the corresponding five-digit seed number S_j. In our example, we have

$$\begin{aligned} n_1 = i_1 S_1 &= \texttt{0227901} \\ n_2 = i_2 S_2 &= \texttt{3967556} \\ n_3 = i_3 S_3 &= \texttt{5106125} \end{aligned}$$

Here the notation $i_j S_j$ means writing the two digits of i_j followed by the five digits of S_j. The precinct number $i = \texttt{023951}$ is apparent by looking at the leading two digits of each number n_j, in sequence.

The second step is to compute the sum L of the square roots of these values:

$$\begin{aligned} L &= \sum_{1 \le j \le b} \sqrt{n_j} \\ &= \sqrt{\texttt{0227901}} \\ &\quad +\sqrt{\texttt{3967556}} \\ &\quad +\sqrt{\texttt{5106125}} \\ &= 477.38977 \\ &\quad 1991.8724 \\ &\quad 2259.6736 \\ &= 4728.9357 \end{aligned}$$

The third and last step is to take the fractional part of L (i.e. the part after the decimal point) as the desired value x_i.

In our example:

$$x_i = f_S(i) = 0.9357 \tag{2}$$

Thus P_i will be audited if and only if $p_i \ge 0.9357$.

Typically, x_i will not be close to p_i, and the comparison is straightforward. In a few cases, x_i might be quite close to p_i, and the issue of roundoff error in the computation of x_i may become relevant. The rare cases when this might be relevant is readily apparent, as x_i and p_i will differ by a small amount (two or less) in the least significant digit position. In such a case, a calculator with higher precision can be used to determine the correct comparison result.

For example, the above values were computed on an inexpensive calculator with an 8-digit display; a ten-digit calculator gives the value:

$$x_i = f_S(i) = 0.935913$$

a high-precision computation (done with the PC program Microsoft Calculator) gives a more accurate result:

$$x_i = 0.935912736408580403646592 19\ldots\ . \tag{3}$$

See the discussion on precision in Section 6.1. We thus see that there is a very small $(0.00021\ldots)$ roundoff error in the computation of x_i on an eight-digit calculator.

The overall computation thus requires typing in three numbers of seven digits each, adding their square roots, and writing down the desired value as the number in the result after the decimal point. The Appendix illustrates key sequences for this computation on a typical calculator.

6 Analysis

This section explores various statistical and other aspects of the SSR function, with the goal of determining if SSR has any weaknesses or vulnerabilities that an adversary could exploit. We find no such weaknesses or vulnerabilities.

We use the general notation introduced above: b is the number of blocks, d is the number of digits in each precinct number block, and k is the number of digits in each seed number. The recommended values for these parameters for the SSR method are $b = 3$, $d = 2$, and $k = 5$.

6.1 Precision

How much precision can one obtain in the computation of $f_S(i)$ using a calculator? Is this enough?

The value L is less than $b \cdot \sqrt{10^{d+k}}$. With our recommended parameter values, $L < 9487$. There are

thus up to four digits before the decimal point in L, so a calculator with an eight-digit display can show four digits of precision after the decimal point, giving four digits of precision in the calculation of f.

Calculators with more precision are readily available. A calculator with a 10-digit display (Casio fx-260 solar) can be bought for $9, a calculator with a 12-digit display (Canon HS-1200TS) can be bought for $14, and a calculator with a 14-digit display (Canon WS-1400H) can be bought for $23, at today's prices; all have square-root functionality.

A calculator with a ten-digit display can show six digits of precision after the decimal point, that is, six digits of precision in the calculation of f, which should be plenty. Similarly, a calculator with a twelve-digit display can show eight digits of accuracy in the computation of f, and a calculator with a fourteen-digit display can show ten digits of accuracy.

If for a given precinct P_i the computed result x_i appears to be equal (or very near, within two in the least significant digit, due to roundoff error, as in the difference between equation (2) and equation (3)) to the precomputed value p_i, one should use a calculator with greater precision to determine if x_i is actually less than p_i, or greater. We can also recommend the use of the PC program "Microsoft Calculator," which has 32 digits of precision, or an equivalent high-precision calculator program on a PC. But this should rarely be necessary (e.g. only five out of every 10,000 precincts might need to do this higher-precision check). The central election officials could also be available by phone to handle inquiries about such "close calls."

6.2 Sensitivity

How sensitive is the value of x_i to changes in the precinct number i or to changes in the seed values S_j?

Since the derivative of the function $\sqrt{z}$ is $1/(2\sqrt{z})$, changing z by an amount a changes $\sqrt{z}$ by an amount approximately equal to $a/(2\sqrt{z})$.

Thus, changing one digit of the precinct number i should change the corresponding square root by at least

$$10^k/(2\sqrt{10^{d+k}}) = \frac{1}{2}10^{(k-d)/2} \ ,$$

using $a \geq 10^k$, and $z \leq 10^{d+k}$, in the previous formula. This lower bound is equal to 15.8 for the standard parameter values ($d = 2$ and $k = 5$). Thus, the change of a single digit of the precinct number will always have a potentially large effect on the computed value of x_i. In general, having $k > d$ should ensure that each digit of the precinct number will have a strong effect on all digits of the output.

Similarly, changing digits of S_j will have similar (but reduced) effects on the final output (reduced effects since the digits of S_j are in less significant positions within n_j).

6.3 Uniformity

We have run chi-square experiments for several fixed values of i (the precinct number) and a large number of different seeds. The results were always that the function f produces results that are very close to uniform. (Bin sizes were 10^{-5}; 10^9 computations of f were performed. A typical χ^2 value obtained was 99506 for 10^5 degrees of freedom; a result at the 44% level for the χ^2 cdf; this is easily within the range of what one would expect for a uniformly distributed random variable.)

6.4 Correlation

We ran some experiments to determine how correlated are values of x_i and x_j when i and j are closely related. This was measured by measuring the average "mod 1 distance" between $f_S(i_1)$ and $f_S(i_2)$ when i_1 and i_2 were closely related. The "mod 1 distance" between x_1 and x_2 is $\min(z, 1-z)$ where $z = |x_1 - x_2|$. The expected value for uncorrelated inputs is 0.250; we observed 0.251.

We ran additional experiments that measured the χ^2 statistic on the pairs $(\lfloor x_1 * 100 \rfloor, \lfloor x_2 * 100 \rfloor)$; these showed little or no correlation (even nonlinear) between the values for different precincts.

None of the results obtained suggest that SSR would be unsuitable for this application.

6.5 Squares

One potential concern is with the proposed SSR method is that some seven-digit numbers are perfect squares, whose square root has a zero fractional part. This could conceivably cause some bias in the SSR output.

However, only one of every 3162 seven-digit numbers is a perfect square, so this effect should be very small. Moreover, the method adds the square roots of three numbers, and the chance that all three of them are perfect squares is very tiny. Thus, the bias caused by the presence of perfect squares seems negligible.

Along the same lines of analysis, we note for the record that if two integers s and t have square roots with the same fractional part, then both fractional parts must be zero (that is, both s and t are square numbers).

7 Variations

7.1 Other parameter values

This SSR method generalizes in the natural manner to handle other values for b, d, and k, instead of the standard recommended values $b = 3$, $d = 2$, $k = 5$.

While we have tested these recommended values for the parameters extensively, we have only lightly tested other parameter settings. Since many potentially attractive alternative parameter settings (e.g. $b = 3$, $d = 3$, $k = 5$, or $b = 1$, $d = 5$, $k = 6$) give worrisome results on statistical tests, we recommend staying with the standard values.

8 Discussion

When using a basic auditing method in an audit there is variability in the number of precincts actually audited, since with a basic auditing method the auditing decision for each precinct is made independently. A uniform threshold of $p = 0.05$ may result in an audit with slightly more, or slightly fewer, than 5% of the precincts being audited. The threshold p might need to be lowered or raised slightly to get the desired number of precincts in the sample.

One could of course use cryptographic hash functions for the pseudo-random function, instead of f. One would concatenate the value of i with the value of S (perhaps with a separator character in between), and use the result as an input to SHA-1 or MD5[3] The output can then be interpreted as a number between 0 and 1, by putting a point before its output. For example, the MD5 hash of the string

$$iS = \texttt{023951-279016755606125}$$

is (in hexadecimal):

$$md5(iS) = \texttt{b2eda26190dbf89f733f946c68305d79}$$

or (interpreting the previous value as base-16 fraction with a point in front, and representing it now in decimal):

$$0.6989\ldots\ .$$

However, this approach seems a bit "heavy-handed" for the present application. It should give good results, but will be a bit more difficult to integrate into election operations, and will be somewhat more difficult for pollworkers and election observers to confirm the determination of the election audit sample.

The approach suggested in this paper allows the central election officials to determine the seed S via dice roll in a public ceremony, for the election officials to publish a list of those precincts to be audited, and for election officials at each precinct, election observers, and even voters to verify the correctness of the list of precincts to be audited. This verification can be performed on a per-precinct basis, using a simple pocket calculator, and/or by checking the whole list of precincts to be audited using independent software.

The use of a pseudo-random function may have additional security advantages, in the sense that it is harder for an adversary to somehow manipulate dice to his advantage. When the dice rolls are used directly to pick precincts as in Cordero et al. [3], a clever adversary might be able to manipulate the

[3]Although the collision-resistance of MD5 has been defeated[10], MD5 still seems fine as a pseudo-random function for applications such as ours. SHA-1 or SHA-256 would of course also be excellent choices.

dice rolls so as to avoid picking any of the corrupted precincts. But when the precincts are picked through the intermediation of a pseudo-random function f, and the dice rolls only determine the seed S for f, it may be much harder for the adversary to determine a seed S (if any exist) that would miss all of the corrupted precincts.

The pseudo-random function of this note may have other uses besides those described in this paper. However, one should not assume that it has any cryptographic strength, or that it would serve satisfactorily in other applications (particularly those in which it is desired to keep the seed secret even though output values are revealed).

Further study is desirable to further understand the strengths and weaknesses of this proposal. But our initial analysis shows no reason to be concerned about using the proposed pseudo-random function f for the application of sampling in election audits. We propose that it is a viable option for this purpose.

9 Related work

There has been significant study of sums of square roots of integers. See, for example, Qian and Wang [9], and Problem 33 of the The Open Problems Project [8]. This problem looks for sums of square roots that are nearly equal to each other. For example:

$$\sqrt{10} + \sqrt{11} - \sqrt{5} - \sqrt{18} \approx 0.0002$$

$$\sqrt{5} + \sqrt{6} + \sqrt{18} - \sqrt{4} - \sqrt{12} - \sqrt{12} \approx 0.000005$$

This research seems to support the general position that working with sums of square roots is rather difficult, in general.

10 Conclusion

We have proposed a simple method for selecting which precincts to audit in a basic post-election (where the precinct audit probabilities have already been determined by some other means). The main virtues of this approach are ease of computation and (thus) ease of verification; the proposed method only requires the evaluation of three square roots. The method should be secure against an adversary's attempts to evade an audit, based on our preliminary examinations, but further study of its security properties is desirable.

Acknowledgments

I'd like to thank Lynn Garland for encouraging me to turn an email on this suggestion into this note, for providing detailed comments on an earlier draft, and for the suggestion to reduce k from 6 (as it was in an earlier draft) to 5 to enable getting 4 digits of output precision on an 8-digit calculator.

References

[1] Javed Aslam, Raluca Popa, and Ronald L. Rivest. On auditing elections when precincts have different sizes, 2008. `http://www.usenix.org/events/evt07/tech/full_papers/aslam/aslam.pdf`.

[2] Joseph A. Calandrino, J. Alex Halderman, and Edward W. Felten. Machine-assisted election auditing. In *Proc. EVT'07*, 2007. `http://www.usenix.org/event/evt07/tech/full_papers/calandrino/calandrino.pdf`.

[3] Arel Cordero, David Wagner, and David Dill. The role of dice in election audits — extended abstract, June 16 2006. IAVoSS Workshop on Trustworthy Elections (WOTE 2006). `http://www.cs.berkeley.edu/~daw/papers/dice-wote06.pdf`.

[4] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *JACM*, 33(4):792–807, Oct 1986. `http://www.wisdom.weizmann.ac.il/~oded/X/ggm.pdf`.

[5] D. Jefferson, E. Ginnold, K. Midstokke, K. Alexander, and A. Lehmkuhl. Post-election audit standards working group report: Evaluation of audit sampling methods and options for strengthening california's manual count, July

2007. http://www.sos.ca.gov/elections/peas/final_peaswg_report.pdf.

[6] John McCarthy, Howard Stanislevic, Mark Lindeman, Arlene Ash, Vittorio Addona, and Mary Batcher. Percentage-based versus SAFE vote tabulation auditing: a graphic comparison. *The American Statistician*, 62(1):11–16, February 2008. Full version: http://verifiedvoting.org/downloads/TAS_paper.pdf.

[7] Lawrence Norden, Aaron Burstein, Joseph Lorenzo Hall, and Margaret Chen. Post-election audits: Restoring trust in elections, August 2007. Brennan Center for Justice at New York University [School of Law and Samuelson Law, Technology and Public Policy Clinic at UC Berkeley School of Law], http://www.brennancenter.org/dynamic/subpages/download_file_50227.pdf.

[8] The Open Problems Project. Problem 33: Sum of square roots, September 2007. http://maven.smith.edu/~orourke/TOPP/P33.html.

[9] Jianbo Qian and Cao An Wang. An optimal bound for sum of square roots of special type of integers. In *The Sixth International Symposium on Operations Research and Its Applications (ISORA'06)*, pages 206–211, August 8–12 2006. http://www.aporc.org/LNOR/6/ISORA2006F15.pdf.

[10] Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In *Proc. Eurocrypt 2005*, pages 19–35. Springer, 2005.

Appendix.

This section illustrates how to compute $f_S(i)$ on a typical calculator. Here we assume that the calculator has a SQRT button and an addition button. The only significant point to note is that the calculator allows one to chain the results of the computation appropriately, so that intermediate results do not need to be copied down or saved in a memory register.

Let the six-digit precinct number i be divided into three two-digit blocks i_2, i_2, i_3.
Let the seed be three five-digit numbers S_1,S_2,S_3.
Clear the calculator.
Enter the two digits of i_1.
Enter the five digits of S_1.
Enter SQRT, then "+"
Enter the two digits of i_2.
Enter the five digits of S_2.
Enter SQRT, then "+"
Enter the two digits of i_3.
Enter the five digits of S_3.
Enter SQRT, then "="
Write down "0."
Write down the digits after decimal point from display.
The result is x_i.

The operations would be similar or identical on other calculators. In addition to simple pocket calculators, one could use Microsoft Calculator on a PC for high precision, since Microsoft Calculator maintains 32 digits of precision. (Note that the "standard" view in this program has a square-root button, while the "scientific" view does not.)