Riffing On (& Ripping Off) Ron

RivestFest
MIT

Philip B. Stark

7 October 2022

University of California, Berkeley

Ron-stats

-  419 publications and talks

Ron-stats

-  419 publications and talks
-  108 relate to elections.

Ron-stats

-  419 publications and talks
-  108 relate to elections.
-  31 articles, proceedings, and chapters

Ron-stats

-  419 publications and talks
-  108 relate to elections.
-  31 articles, proceedings, and chapters
-  10 of those joint w PBS; 25 joint total; 7 refereed; my most frequent collaborator

Ron-stats

-  419 publications and talks
-  108 relate to elections.
-  31 articles, proceedings, and chapters
-  10 of those joint w PBS; 25 joint total; 7 refereed; my most frequent collaborator

Today, focus on things that aren't joint work.

Lessons from working with Ron:

- Simple. Practical. Understandable. Communicate/explain/teach.

Lessons from working with Ron:

- Simple. Practical. Understandable. Communicate/explain/teach.
- Frolic in solution space. It helps understand the problem and the constraints, and may lead to a solution.

Lessons from working with Ron:

- Simple. Practical. Understandable. Communicate/explain/teach.
- Frolic in solution space. It helps understand the problem and the constraints, and may lead to a solution.
- Consider changing the problem. What's the real goal? What are the real constraints?

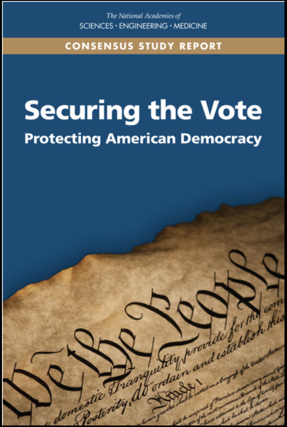
Lessons from working with Ron:

- Simple. Practical. Understandable. Communicate/explain/teach.
- Frolic in solution space. It helps understand the problem and the constraints, and may lead to a solution.
- Consider changing the problem. What's the real goal? What are the real constraints?
- Emphasize principles over technique—but implement it, and algorithms matter!

Some of Ron's election topics

- Public service
- Verifiability
- Verification and Auditing
- Voting systems
- Bespoke social choice functions
- Whimsical/Pedagogical

TGDC, Testimony to House Administration (x2); Testimony to Presidential Commission on Election Admin; Verified Voting Board of Directors; open-source software



COMMITTEE ON THE FUTURE OF VOTING:
ACCESSIBLE, RELIABLE, VERIFABLE TECHNOLOGY

Co-chairs

LEE C. BOLLINGER, President, Columbia University
MICHAEL A. McROBBIE, President, Indiana University

Members

ANDREW W. APPEL, Eugene Higgins Professor of Computer Science, Princeton University
JOSH BENALOH, Senior Cryptographer, Microsoft Research
KAREN COOK (NAS), Ray Lyman Wilbur Professor of Sociology; Director of the Institute for Research in the Social Sciences (IRiSS); and Vice-Provost for Faculty Development and Diversity, Stanford University
DANA DeBEAUVOIR, Travis County Clerk, County of Travis, TX
MOON DUCHIN, Associate Professor of Mathematics and Founding Director, Program in Science, Technology, and Society, Tufts University
JUAN E. GILBERT, Andrew Banks Family Preeminence Endowed Professor and Chair of the Computer and Information Science and Engineering Department, University of Florida
SUSAN L. GRAHAM (NAE), Pehong Chen Distinguished Professor Emerita, Computer Science Division, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley
NEAL KELLEY, Registrar of Voters and Chief of Elections, County of Orange, CA
KEVIN J. KENNEDY, Director and General Counsel (retired), Wisconsin Government Accountability Board
NATHANIEL PERSILY, James B. McClatchy Professor of Law, Stanford Law School
RONALD L. RIVEST (NAS/NAE), Institute Professor, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology
CHARLES STEWART III, Kenan Sahin Distinguished Professor of Political Science, Massachusetts Institute of Technology

Verifiability: Software independence

PHILOSOPHICAL TRANSACTIONS OF THE ROYAL SOCIETY A

Phil. Trans. R. Soc. A (2008) **366**, 3759–3767
doi:10.1098/rsta.2008.0149
Published online 6 August 2008

On the notion of 'software independence' in voting systems

By Ronald L. Rivest[1,*]

[1]*Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology (MIT), Cambridge, MA 02139, USA*

This paper defines and explores the notion of 'software independence' in voting systems: 'A voting system is *software independent* if an (undetected) change or error in its software cannot cause an undetectable change or error in an election outcome'. For example, optical scan and some cryptographically based voting systems are software independent. Variations and implications of this definition are explored. It is proposed that software-independent voting systems should be preferred, and *software-dependent* voting systems should be avoided.

An initial version of this paper was prepared for use by the Technical Guidelines Development Committee in their development of the Voluntary Voting System Guidelines, which will specify the requirements that the USA voting systems must meet to receive certification.

Keywords: security; voting; software independence

1. Introduction

The main purpose of this paper is to introduce and carefully define the terminology of 'software-independent' and 'software-dependent' voting systems, and to discuss their properties. This paper is definitional in character; there are

(*a*) *Refinements and elaborations of software independence*

There are a number of possible refinements and elaborations of the notion of software independence. I now motivate and introduce the distinction between *strong software independence* and *weak software independence*.

Security mechanisms are typically one of two forms: *prevention* or *detection*. Detection mechanisms may also be coupled with means for *recovery*. When identification of participants and accountability for actions is also present, then detection mechanisms are also the foundation for *deterrence*. Given the importance of recovery mechanisms in addition to detection mechanisms, I propose two definitions that are as follows.

— A voting system is *strongly software independent* if an (undetected) change or error in its software cannot cause an undetectable change or error in an election outcome, and, moreover, a detected change or error in an election outcome (due to a change or error in the software) can be corrected without rerunning the election.
— A voting system that is *weakly software independent* conforms to the basic definition of software independence; that is, there is no recovery mechanism.

Voluntary Voting System Guidelines VVSG 2.0

Requirements for the Voluntary Voting System Guidelines 2.0

February 10, 2021

Prepared for the ***Election Assistance Commission***

At the direction of the
Technical Guidelines Development Committee

Principle 9: AUDITABLE

The voting system is auditable and enables evidence-based elections.

9.1 - An error or fault in the voting system software or hardware cannot cause an undetectable change in election results.

9.2 - The voting system produces readily available records that provide the ability to check whether the election outcome is correct and, to the extent possible, identify the root cause of any irregularities.

9.3 - Voting system records are resilient in the presence of intentional forms of tampering and accidental errors.

9.4 - The voting system supports efficient audits.

- If a system is not SI, it can change results with no trace. Example: DREs

Voluntary Voting System Guidelines VVSG 2.0

Requirements for the Voluntary Voting System Guidelines 2.0

February 10, 2021

Prepared for the ***Election Assistance Commission***

At the direction of the
Technical Guidelines Development Committee

Principle 9: AUDITABLE

The voting system is auditable and enables evidence-based elections.

> 9.1 - An error or fault in the voting system software or hardware cannot cause an undetectable change in election results.

9.2 - The voting system produces readily available records that provide the ability to check whether the election outcome is correct and, to the extent possible, identify the root cause of any irregularities.

9.3 - Voting system records are resilient in the presence of intentional forms of tampering and accidental errors.

9.4 - The voting system supports efficient audits.

- If a system is not SI, it can change results with no trace. Example: DREs
- If a system is not SSI, even if a failure is "detected," recovery might be impossible. Example: BMDs

Voluntary Voting System Guidelines VVSG 2.0

Requirements for the Voluntary Voting System Guidelines 2.0

February 10, 2021

Prepared for the ***Election Assistance Commission***

At the direction of the ***Technical Guidelines Development Committee***

Principle 9: AUDITABLE

The voting system is auditable and enables evidence-based elections.

9.1 - An error or fault in the voting system software or hardware cannot cause an undetectable change in election results.

9.2 - The voting system produces readily available records that provide the ability to check whether the election outcome is correct and, to the extent possible, identify the root cause of any irregularities.

9.3 - Voting system records are resilient in the presence of intentional forms of tampering and accidental errors.

9.4 - The voting system supports efficient audits.

- If a system is not SI, it can change results with no trace. Example: DREs
- If a system is not SSI, even if a failure is "detected," recovery might be impossible. Example: BMDs
- SI is a necessary security property for voting systems.

Voluntary Voting System Guidelines VVSG 2.0

Requirements for the Voluntary Voting System Guidelines 2.0

February 10, 2021

Prepared for the ***Election Assistance Commission***

At the direction of the
Technical Guidelines Development Committee

Principle 9: AUDITABLE

The voting system is auditable and enables evidence-based elections.

9.1 - An error or fault in the voting system software or hardware cannot cause an undetectable change in election results.

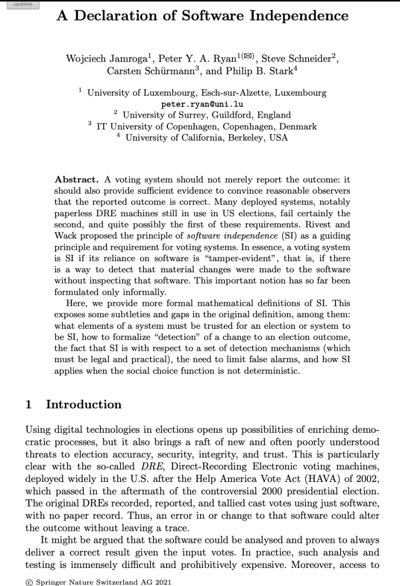
9.2 - The voting system produces readily available records that provide the ability to check whether the election outcome is correct and, to the extent possible, identify the root cause of any irregularities.

9.3 - Voting system records are resilient in the presence of intentional forms of tampering and accidental errors.

9.4 - The voting system supports efficient audits.

- If a system is not SI, it can change results with no trace. Example: DREs
- If a system is not SSI, even if a failure is “detected,” recovery might be impossible. Example: BMDs
- SI is a necessary security property for voting systems.
- NB: *principle* not *technology* or *technique*

Riffing on Ron: Formalizing SI



updates

A Declaration of Software Independence

Wojciech Jamroga[1], Peter Y. A. Ryan[1(✉)], Steve Schneider[2], Carsten Schürmann[3], and Philip B. Stark[4]

[1] University of Luxembourg, Esch-sur-Alzette, Luxembourg
peter.ryan@uni.lu
[2] University of Surrey, Guildford, England
[3] IT University of Copenhagen, Copenhagen, Denmark
[4] University of California, Berkeley, USA

Abstract. A voting system should not merely report the outcome: it should also provide sufficient evidence to convince reasonable observers that the reported outcome is correct. Many deployed systems, notably paperless DRE machines still in use in US elections, fail certainly the second, and quite possibly the first of these requirements. Rivest and Wack proposed the principle of *software independence* (SI) as a guiding principle and requirement for voting systems. In essence, a voting system is SI if its reliance on software is "tamper-evident", that is, if there is a way to detect that material changes were made to the software without inspecting that software. This important notion has so far been formulated only informally.

Here, we provide more formal mathematical definitions of SI. This exposes some subtleties and gaps in the original definition, among them: what elements of a system must be trusted for an election or system to be SI, how to formalize "detection" of a change to an election outcome, the fact that SI is with respect to a set of detection mechanisms (which must be legal and practical), the need to limit false alarms, and how SI applies when the social choice function is not deterministic.

1 Introduction

Using digital technologies in elections opens up possibilities of enriching democratic processes, but it also brings a raft of new and often poorly understood threats to election accuracy, security, integrity, and trust. This is particularly clear with the so-called *DRE*, Direct-Recording Electronic voting machines, deployed widely in the U.S. after the Help America Vote Act (HAVA) of 2002, which passed in the aftermath of the controversial 2000 presidential election. The original DREs recorded, reported, and tallied cast votes using just software, with no paper record. Thus, an error in or change to that software could alter the outcome without leaving a trace.

It might be argued that the software could be analysed and proven to always deliver a correct result given the input votes. In practice, such analysis and testing is immensely difficult and prohibitively expensive. Moreover, access to

- Is SI a property of system or a run of the system?

Riffing on Ron: Formalizing SI

updates

A Declaration of Software Independence

Wojciech Jamroga[1], Peter Y. A. Ryan[1(✉)], Steve Schneider[2], Carsten Schürmann[3], and Philip B. Stark[4]

[1] University of Luxembourg, Esch-sur-Alzette, Luxembourg
peter.ryan@uni.lu
[2] University of Surrey, Guildford, England
[3] IT University of Copenhagen, Copenhagen, Denmark
[4] University of California, Berkeley, USA

Abstract. A voting system should not merely report the outcome: it should also provide sufficient evidence to convince reasonable observers that the reported outcome is correct. Many deployed systems, notably paperless DRE machines still in use in US elections, fail certainly the second, and quite possibly the first of these requirements. Rivest and Wack proposed the principle of *software independence* (SI) as a guiding principle and requirement for voting systems. In essence, a voting system is SI if its reliance on software is "tamper-evident", that is, if there is a way to detect that material changes were made to the software without inspecting that software. This important notion has so far been formulated only informally.

Here, we provide more formal mathematical definitions of SI. This exposes some subtleties and gaps in the original definition, among them: what elements of a system must be trusted for an election or system to be SI, how to formalize "detection" of a change to an election outcome, the fact that SI is with respect to a set of detection mechanisms (which must be legal and practical), the need to limit false alarms, and how SI applies when the social choice function is not deterministic.

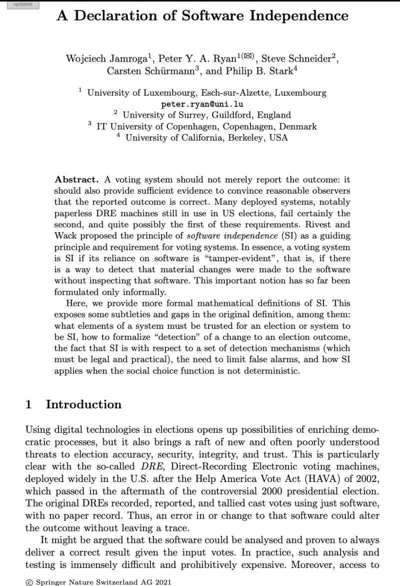
1 Introduction

Using digital technologies in elections opens up possibilities of enriching democratic processes, but it also brings a raft of new and often poorly understood threats to election accuracy, security, integrity, and trust. This is particularly clear with the so-called *DRE*, Direct-Recording Electronic voting machines, deployed widely in the U.S. after the Help America Vote Act (HAVA) of 2002, which passed in the aftermath of the controversial 2000 presidential election. The original DREs recorded, reported, and tallied cast votes using just software, with no paper record. Thus, an error in or change to that software could alter the outcome without leaving a trace.

It might be argued that the software could be analysed and proven to always deliver a correct result given the input votes. In practice, such analysis and testing is immensely difficult and prohibitively expensive. Moreover, access to

- Is SI a property of system or a run of the system?
- What does "detectable" mean? Are there any constraints on the cost of the detection? What if the only method of detection is infeasible, illegal, or violates voter privacy?

Riffing on Ron: Formalizing SI



A Declaration of Software Independence

Wojciech Jamroga[1], Peter Y. A. Ryan[1(✉)], Steve Schneider[2], Carsten Schürmann[3], and Philip B. Stark[4]

[1] University of Luxembourg, Esch-sur-Alzette, Luxembourg
peter.ryan@uni.lu
[2] University of Surrey, Guildford, England
[3] IT University of Copenhagen, Copenhagen, Denmark
[4] University of California, Berkeley, USA

Abstract. A voting system should not merely report the outcome: it should also provide sufficient evidence to convince reasonable observers that the reported outcome is correct. Many deployed systems, notably paperless DRE machines still in use in US elections, fail certainly the second, and quite possibly the first of these requirements. Rivest and Wack proposed the principle of *software independence* (SI) as a guiding principle and requirement for voting systems. In essence, a voting system is SI if its reliance on software is "tamper-evident", that is, if there is a way to detect that material changes were made to the software without inspecting that software. This important notion has so far been formulated only informally.

Here, we provide more formal mathematical definitions of SI. This exposes some subtleties and gaps in the original definition, among them: what elements of a system must be trusted for an election or system to be SI, how to formalize "detection" of a change to an election outcome, the fact that SI is with respect to a set of detection mechanisms (which must be legal and practical), the need to limit false alarms, and how SI applies when the social choice function is not deterministic.

1 Introduction

Using digital technologies in elections opens up possibilities of enriching democratic processes, but it also brings a raft of new and often poorly understood threats to election accuracy, security, integrity, and trust. This is particularly clear with the so-called *DRE*, Direct-Recording Electronic voting machines, deployed widely in the U.S. after the Help America Vote Act (HAVA) of 2002, which passed in the aftermath of the controversial 2000 presidential election. The original DREs recorded, reported, and tallied cast votes using just software, with no paper record. Thus, an error in or change to that software could alter the outcome without leaving a trace.

It might be argued that the software could be analysed and proven to always deliver a correct result given the input votes. In practice, such analysis and testing is immensely difficult and prohibitively expensive. Moreover, access to

- Is SI a property of system or a run of the system?
- What does "detectable" mean? Are there any constraints on the cost of the detection? What if the only method of detection is infeasible, illegal, or violates voter privacy?
- Who or what does the detection?

Riffing on Ron: Formalizing SI

updates

A Declaration of Software Independence

Wojciech Jamroga[1], Peter Y. A. Ryan[1(✉)], Steve Schneider[2], Carsten Schürmann[3], and Philip B. Stark[4]

[1] University of Luxembourg, Esch-sur-Alzette, Luxembourg
peter.ryan@uni.lu
[2] University of Surrey, Guildford, England
[3] IT University of Copenhagen, Copenhagen, Denmark
[4] University of California, Berkeley, USA

Abstract. A voting system should not merely report the outcome: it should also provide sufficient evidence to convince reasonable observers that the reported outcome is correct. Many deployed systems, notably paperless DRE machines still in use in US elections, fail certainly the second, and quite possibly the first of these requirements. Rivest and Wack proposed the principle of *software independence* (SI) as a guiding principle and requirement for voting systems. In essence, a voting system is SI if its reliance on software is "tamper-evident", that is, if there is a way to detect that material changes were made to the software without inspecting that software. This important notion has so far been formulated only informally.

Here, we provide more formal mathematical definitions of SI. This exposes some subtleties and gaps in the original definition, among them: what elements of a system must be trusted for an election or system to be SI, how to formalize "detection" of a change to an election outcome, the fact that SI is with respect to a set of detection mechanisms (which must be legal and practical), the need to limit false alarms, and how SI applies when the social choice function is not deterministic.

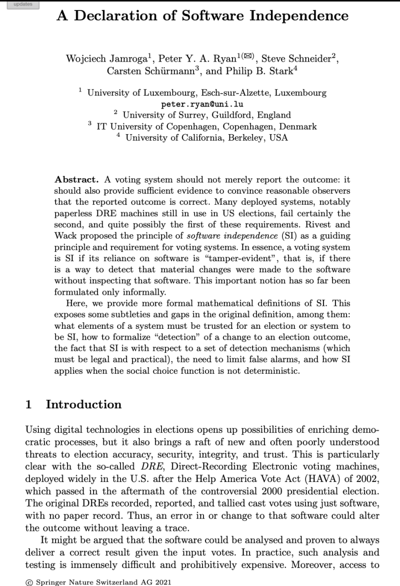
1 Introduction

Using digital technologies in elections opens up possibilities of enriching democratic processes, but it also brings a raft of new and often poorly understood threats to election accuracy, security, integrity, and trust. This is particularly clear with the so-called *DRE*, Direct-Recording Electronic voting machines, deployed widely in the U.S. after the Help America Vote Act (HAVA) of 2002, which passed in the aftermath of the controversial 2000 presidential election. The original DREs recorded, reported, and tallied cast votes using just software, with no paper record. Thus, an error in or change to that software could alter the outcome without leaving a trace.

It might be argued that the software could be analysed and proven to always deliver a correct result given the input votes. In practice, such analysis and testing is immensely difficult and prohibitively expensive. Moreover, access to

- Is SI a property of system or a run of the system?
- What does "detectable" mean? Are there any constraints on the cost of the detection? What if the only method of detection is infeasible, illegal, or violates voter privacy?
- Who or what does the detection?
- Is there any penalty for false alarms?

Riffing on Ron: Formalizing SI



updates

A Declaration of Software Independence

Wojciech Jamroga[1], Peter Y. A. Ryan[1(✉)], Steve Schneider[2], Carsten Schürmann[3], and Philip B. Stark[4]

[1] University of Luxembourg, Esch-sur-Alzette, Luxembourg
peter.ryan@uni.lu
[2] University of Surrey, Guildford, England
[3] IT University of Copenhagen, Copenhagen, Denmark
[4] University of California, Berkeley, USA

Abstract. A voting system should not merely report the outcome: it should also provide sufficient evidence to convince reasonable observers that the reported outcome is correct. Many deployed systems, notably paperless DRE machines still in use in US elections, fail certainly the second, and quite possibly the first of these requirements. Rivest and Wack proposed the principle of *software independence* (SI) as a guiding principle and requirement for voting systems. In essence, a voting system is SI if its reliance on software is "tamper-evident", that is, if there is a way to detect that material changes were made to the software without inspecting that software. This important notion has so far been formulated only informally.

Here, we provide more formal mathematical definitions of SI. This exposes some subtleties and gaps in the original definition, among them: what elements of a system must be trusted for an election or system to be SI, how to formalize "detection" of a change to an election outcome, the fact that SI is with respect to a set of detection mechanisms (which must be legal and practical), the need to limit false alarms, and how SI applies when the social choice function is not deterministic.

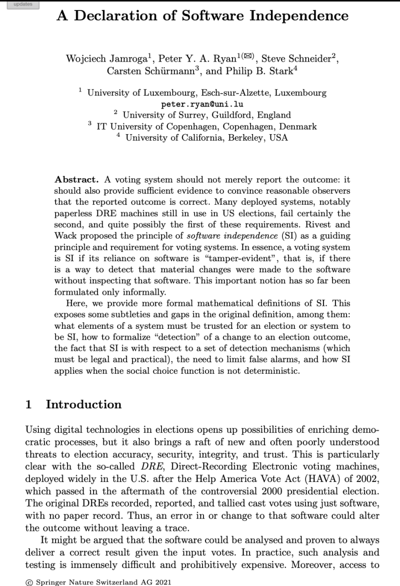
1 Introduction

Using digital technologies in elections opens up possibilities of enriching democratic processes, but it also brings a raft of new and often poorly understood threats to election accuracy, security, integrity, and trust. This is particularly clear with the so-called *DRE*, Direct-Recording Electronic voting machines, deployed widely in the U.S. after the Help America Vote Act (HAVA) of 2002, which passed in the aftermath of the controversial 2000 presidential election. The original DREs recorded, reported, and tallied cast votes using just software, with no paper record. Thus, an error in or change to that software could alter the outcome without leaving a trace.

It might be argued that the software could be analysed and proven to always deliver a correct result given the input votes. In practice, such analysis and testing is immensely difficult and prohibitively expensive. Moreover, access to

- Is SI a property of system or a run of the system?
- What does "detectable" mean? Are there any constraints on the cost of the detection? What if the only method of detection is infeasible, illegal, or violates voter privacy?
- Who or what does the detection?
- Is there any penalty for false alarms?
- Does a system that always sounds an alarm "detect" problems?

Riffing on Ron: Formalizing SI



updates

A Declaration of Software Independence

Wojciech Jamroga[1], Peter Y. A. Ryan[1(✉)], Steve Schneider[2], Carsten Schürmann[3], and Philip B. Stark[4]

[1] University of Luxembourg, Esch-sur-Alzette, Luxembourg
peter.ryan@uni.lu
[2] University of Surrey, Guildford, England
[3] IT University of Copenhagen, Copenhagen, Denmark
[4] University of California, Berkeley, USA

Abstract. A voting system should not merely report the outcome: it should also provide sufficient evidence to convince reasonable observers that the reported outcome is correct. Many deployed systems, notably paperless DRE machines still in use in US elections, fail certainly the second, and quite possibly the first of these requirements. Rivest and Wack proposed the principle of *software independence* (SI) as a guiding principle and requirement for voting systems. In essence, a voting system is SI if its reliance on software is "tamper-evident", that is, if there is a way to detect that material changes were made to the software without inspecting that software. This important notion has so far been formulated only informally.

Here, we provide more formal mathematical definitions of SI. This exposes some subtleties and gaps in the original definition, among them: what elements of a system must be trusted for an election or system to be SI, how to formalize "detection" of a change to an election outcome, the fact that SI is with respect to a set of detection mechanisms (which must be legal and practical), the need to limit false alarms, and how SI applies when the social choice function is not deterministic.

1 Introduction

Using digital technologies in elections opens up possibilities of enriching democratic processes, but it also brings a raft of new and often poorly understood threats to election accuracy, security, integrity, and trust. This is particularly clear with the so-called *DRE*, Direct-Recording Electronic voting machines, deployed widely in the U.S. after the Help America Vote Act (HAVA) of 2002, which passed in the aftermath of the controversial 2000 presidential election. The original DREs recorded, reported, and tallied cast votes using just software, with no paper record. Thus, an error in or change to that software could alter the outcome without leaving a trace.

It might be argued that the software could be analysed and proven to always deliver a correct result given the input votes. In practice, such analysis and testing is immensely difficult and prohibitively expensive. Moreover, access to

- Is SI a property of system or a run of the system?
- What does "detectable" mean? Are there any constraints on the cost of the detection? What if the only method of detection is infeasible, illegal, or violates voter privacy?
- Who or what does the detection?
- Is there any penalty for false alarms?
- Does a system that always sounds an alarm "detect" problems?
- Is there a dispute-resolution mechanism? Vulnerability to FUD attacks?

Riffing on Ron: More stringent requirements

Election Law Journal: Rules, Politics, and Policy > VOL. 19, NO. 3 | Original Research Articles normal

Ballot-Marking Devices Cannot Ensure the Will of the Voters

Andrew W. Appel, Richard A. DeMillo, and Philip B. Stark

Published Online: 17 Sep 2020 | https://doi-org.libproxy.berkeley.edu/10.1089/elj.2019.0619

Permissions & Citations Share

Abstract

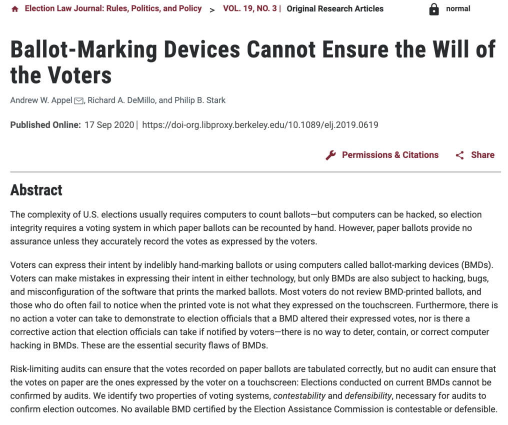
The complexity of U.S. elections usually requires computers to count ballots—but computers can be hacked, so election integrity requires a voting system in which paper ballots can be recounted by hand. However, paper ballots provide no assurance unless they accurately record the votes as expressed by the voters.

Voters can express their intent by indelibly hand-marking ballots or using computers called ballot-marking devices (BMDs). Voters can make mistakes in expressing their intent in either technology, but only BMDs are also subject to hacking, bugs, and misconfiguration of the software that prints the marked ballots. Most voters do not review BMD-printed ballots, and those who do often fail to notice when the printed vote is not what they expressed on the touchscreen. Furthermore, there is no action a voter can take to demonstrate to election officials that a BMD altered their expressed votes, nor is there a corrective action that election officials can take if notified by voters—there is no way to deter, contain, or correct computer hacking in BMDs. These are the essential security flaws of BMDs.

Risk-limiting audits can ensure that the votes recorded on paper ballots are tabulated correctly, but no audit can ensure that the votes on paper are the ones expressed by the voter on a touchscreen: Elections conducted on current BMDs cannot be confirmed by audits. We identify two properties of voting systems, *contestability* and *defensibility*, necessary for audits to confirm election outcomes. No available BMD certified by the Election Assistance Commission is contestable or defensible.

- A system is *contestible* if an (undetected) change or error in its software that causes a change or error in an election outcome can always produce public evidence that the outcome is untrustworthy.

Riffing on Ron: More stringent requirements


Election Law Journal: Rules, Politics, and Policy > VOL. 19, NO. 3 | Original Research Articles normal

Ballot-Marking Devices Cannot Ensure the Will of the Voters

Andrew W. Appel, Richard A. DeMillo, and Philip B. Stark

Published Online: 17 Sep 2020 | https://doi-org.libproxy.berkeley.edu/10.1089/elj.2019.0619

Permissions & Citations Share

Abstract

The complexity of U.S. elections usually requires computers to count ballots—but computers can be hacked, so election integrity requires a voting system in which paper ballots can be recounted by hand. However, paper ballots provide no assurance unless they accurately record the votes as expressed by the voters.

Voters can express their intent by indelibly hand-marking ballots or using computers called ballot-marking devices (BMDs). Voters can make mistakes in expressing their intent in either technology, but only BMDs are also subject to hacking, bugs, and misconfiguration of the software that prints the marked ballots. Most voters do not review BMD-printed ballots, and those who do often fail to notice when the printed vote is not what they expressed on the touchscreen. Furthermore, there is no action a voter can take to demonstrate to election officials that a BMD altered their expressed votes, nor is there a corrective action that election officials can take if notified by voters—there is no way to deter, contain, or correct computer hacking in BMDs. These are the essential security flaws of BMDs.

Risk-limiting audits can ensure that the votes recorded on paper ballots are tabulated correctly, but no audit can ensure that the votes on paper are the ones expressed by the voter on a touchscreen: Elections conducted on current BMDs cannot be confirmed by audits. We identify two properties of voting systems, *contestability* and *defensibility*, necessary for audits to confirm election outcomes. No available BMD certified by the Election Assistance Commission is contestable or defensible.

- A system is *contestible* if an (undetected) change or error in its software that causes a change or error in an election outcome can always produce public evidence that the outcome is untrustworthy.
- A system is *defensible* if, when the reported outcome is correct, it is possible to generate convincing public evidence that the reported outcome is correct—despite any malfunctions, software errors, or software alterations that might have occurred.

Riffing on Ron: What if nobody looks? Evidence-based elections

Evidence-Based Elections

Philip B. Stark and David Wagner | University of California, Berkeley

Elections should be structured to provide convincing affirmative evidence that the reported outcomes actually reflect how people voted. This can be accomplished with a combination of software-independent voting systems, compliance audits, and risk-limiting audits.

Ideally, what should an election do? Certainly, it should find out who won, but we believe it also should produce convincing evidence that it found the real winners—or report that it can't. This isn't automatic; it requires thoughtful design of voting equipment, carefully planned and implemented voting and vote-counting processes, and rigorous postelection auditing.

The systems and processes currently deployed in the US often fail to meet this goal, due to shortcomings of the equipment, gaps in processes, and failures to audit effectively. The first essential requirement is voting equipment that produces a trustworthy audit trail that can be used to confirm that the votes were recorded and tabulated correctly. Given the present state of technology, this means the voting system must produce a tangible, physical record of the vote that can be checked by the voter for accuracy and retained for auditing purposes—typically, a voter-verifiable paper record (VVPR).

Currently, approximately 25 percent of US voters use paperless electronic voting machines that don't produce such a record.[1]

Because paperless electronic voting machines rely on complex software and hardware, and because there's no feasible way to ensure that the voting software is free of bugs or that the hardware is executing the proper software, there's no guarantee that electronic voting machines record votes accurately. And, because paperless voting machines preserve only an electronic record of the vote that can't be directly observed by voters, there's no way to produce convincing evidence that the electronic record accurately reflects voters' intent. Internet voting shares the shortcomings of paperless electronic voting machines and has additional vulnerabilities.

Numerous electronic voting equipment failures have been documented. Paperless voting machines in Carteret County, North Carolina, irretrievably lost 4,400 votes; other machines in Mecklenburg, North Carolina, recorded 3,955 more votes than the number of people who voted; in Bernalillo County, New Mexico, machines recorded 2,700 more votes than voters; in Mahoning County, Ohio, some machines reported a negative total vote count; and in Fairfax, Virginia, county officials found that for every 100 or so votes cast for one candidate, the electronic voting machines subtracted one vote for her.[2] In short, when elections are conducted on paperless voting machines, there's no way to produce convincing evidence that the right candidate won.

VVPRs are important, but they aren't a panacea. If these records aren't examined after the election, then their value is eliminated. For instance, in 13 states, a

Evidence-Based Elections: Create a Meaningful Paper Trail, then Audit

Andrew W. Appel (Princeton University)
Philip B. Stark (University of California, Berkeley)

Evidence-Based Elections

There is no perfect, infallible way to count votes. All methods—including optical scan, touchscreen, and hand counting—are subject to errors, procedural lapses, and deliberate manipulation. Almost all U.S. jurisdictions count their votes using computer-based technology, such as touchscreens and optical-scan machines. Computer-based methods are subject to "hacking", that is, the replacement of legitimate vote-counting software with a computer program that changes (some fraction of) the votes in favor of the hacker's preferred party. Hacking can be performed remotely (even if the machines are supposedly "never connected to the Internet") and it is very difficult to detect. Voters and election administrators see nothing out of the ordinary.

The vulnerability of computers to hacking is well understood. Modern computer systems, including voting machines, have many layers of software, comprising millions of lines of computer code; there are thousands of bugs in that code.[123] Some of those bugs are security vulnerabilities that permit attackers to modify or replace the software in the upper layers; so we can never be sure that the legitimate vote-counting software or the vote-marking user interface is actually the software running on election day.[4]

One might think, "our voting machines are never connected to the Internet, so hackers cannot get to them." But all voting machines need to be programmed for each new election: they need a "ballot-definition file" with the contests and candidate names for each election, and lists of the contests different voters are eligible to vote in. This programming is typically done via removable media such as a USB thumbdrive or a memory card. Vote-stealing malware can

Verification: Detection Audits

Percentage-Based versus Statistical-Power-Based Vote Tabulation Audits

John McCarthy, Howard Stanislevic, Mark Lindeman, Arlene S. Ash, Vittorio Addona, and Mary Batcher

Several pending federal and state electoral-integrity bills specify hand audits of 1% to 10% of all precincts. However, percentage-based audits are usually inefficient, because they require large samples for large jurisdictions, even though the sample needed to achieve good accuracy is much more affected by the closeness of the contest than population size. Percentage-based audits can also be ineffective, since close contests may require auditing a large fraction of the total to provide confidence in the outcome. We present a plausible statistical framework that we have used in advising state and local election officials and legislators. In recent federal elections, this audit model would have required approximately the same effort and resources as the less effective percentage-based audits now being considered.

KEY WORDS: Election audits; Election recounts; Electronic voting; Precinct sampling.

1. INTRODUCTION

Electronic vote tally miscounts arise for many reasons, including hardware malfunctions, unintentional programming errors, malicious tampering, or stray ballot marks that interfere with correct counting. Thus, Congress and several states are considering requiring audits to compare machine tabulations with hand counts of paper ballots in randomly chosen precincts. Audits should be highly effective in detecting miscounts large enough to alter election outcomes; and they should be efficient—no larger than necessary to confirm the winners. While financial and quality control audits set sample sizes that are very likely to detect errors large enough to cause harm, most proposed election auditing laws specify sampling fixed or tiered percentages of precincts. For example, Connecticut has just adopted a law (Public Act 07-194) requiring random audits of 10% of voting districts (precincts) in selected contests. We believe that the laws are written this way because most nonstatisticians have unrealistic fears about the inadequacy of small-percentage audits; because the authors have not measured the statistical effectiveness of percentage-based schemes in general; and because statisticians have—thus far—rarely been involved in drafting audit options for legislators. Statisticians of course know that we can measure the effectiveness of sampling strategies by their statistical power, which principally depends on the number of units sampled and the size of the effect to be detected. Thus, fixed-percentage audits are inefficient (too large) in the vast majority of contests, especially in statewide contests that involve many hundreds of precincts and that are not close; also, they are ineffective (too small) in the rare contests with small winning margins. However, most statisticians know little about election procedures and are not well-equipped to respond when asked "what percentage shall we put in the bill?" We hope that this article helps fill that gap.

Vote tabulation audits entail supervised hand-to-eye manual counts of all voter-verified paper ballots in a subset of precincts, randomly selected shortly after an election and before results are certified. We assume that a "hard copy" record of each voter's choice, one that was reviewable by the voter

On Auditing Elections When Precincts Have Different Sizes

Javed A. Aslam
College of Computer and Information Science
Northeastern University
Boston, MA 02115
jaa@ccs.neu.edu

Raluca A. Popa and Ronald L. Rivest
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139
{ralucap, rivest}@mit.edu

Abstract

We address the problem of auditing an election when precincts may have different sizes. Prior work in this field has emphasized the simpler case when all precincts have the same size. Using auditing methods developed for use with equal-sized precincts can, however, be inefficient or result in loss of statistical confidence when applied to elections with variable-sized precincts.

We survey, evaluate, and compare a variety of approaches to the variable-sized precinct auditing problem, including the SAFE method [11] which is based on theory developed for equal-sized precincts. We introduce new methods such as the negative-exponential method "NEGEXP" that select precincts independently for auditing with predetermined probabilities, and the "PPEBWR" method that uses a sequence of rounds to select precincts with replacement according to some predetermined probability distribution that may depend on error bounds for each precinct (hence the name PPEBWR: probability proportional to error bounds, with replacement), where the error bounds may depend on the sizes of the precincts, or on how the votes were cast in each precinct.

We give experimental results showing that NEGEXP and PPEBWR can *dramatically* reduce (by a factor or two or three) the cost of auditing compared to methods such as SAFE that depend on the use of uniform sampling. Sampling so that larger precincts are audited with appro-

1 Introduction

Post-election audits are an essential tool for ensuring the integrity of election outcomes. They can detect, with high probability, both errors due to machine misprogramming and errors due to malicious manipulation of electronic vote totals. By using statistical samples, they are quite efficient and economical. This paper explores auditing approaches that achieve improved efficiency (sometimes by a factor of two or three, measured in terms of the number of votes counted) over previous methods.

Suppose we have an election with n precincts, $P_1, \ldots, P_n$. Let v_i denote the number of voters who voted in precinct P_i; we call v_i the "size" of the precinct P_i. Let the total number of such voters be $V = \sum_i v_i$. Assume without loss of generality that $v_1 \geq v_2 \geq \cdots \geq v_n$.

We focus on auditing precincts as opposed to votes because this is the common form of auditing encountered in practice. If one is interested in sampling votes, then the results in Aslam et al. [1] apply because the votes can be modeled as precincts of equal size (in particular, of size one). In this paper, we are interested in the more general problem, that is, when precincts have different sizes.

Precinct sizes can vary dramatically, sometimes by an order of magnitude or more. See Figure 2. Methods for auditing elections must, if they are to be efficient and effective, take such precinct size variations into account.

Suppose further that in precinct P_i we have both electronic records and paper records for each voter. The

Riffing on Ron: from detect problems to affirmative evidence

The Annals of Applied Statistics
2008, Vol. 2, No. 2, 550–581
DOI: 10.1214/08-AOAS161

CONSERVATIVE STATISTICAL POST-ELECTION AUDITS

BY PHILIP B. STARK

University of California, Berkeley

There are many sources of error in counting votes: the apparent winner might not be the rightful winner. Hand tallies of the votes in a random sample of precincts can be used to test the hypothesis that a full manual recount would find a different outcome. This paper develops a conservative sequential test based on the vote-counting errors found in a hand tally of a simple or stratified random sample of precincts. The procedure includes a natural escalation: If the hypothesis that the apparent outcome is incorrect is not rejected at stage s, more precincts are audited. Eventually, either the hypothesis is rejected—and the apparent outcome is confirmed—or all precincts have been audited and the true outcome is known. The test uses a priori bounds on the overstatement of the margin that could result from error in each precinct. Such bounds can be derived from the reported counts in each precinct and upper bounds on the number of votes cast in each precinct. The test allows errors in different precincts to be treated differently to reflect voting technology or precinct sizes. It is not optimal, but it is conservative: the chance of erroneously confirming the outcome of a contest if a full manual recount would show a different outcome is no larger than the nominal significance level. The approach also gives a conservative P-value for the hypothesis that a full manual recount would find a different outcome, given the errors found in a fixed size sample. This is illustrated with two contests from November, 2006: the U.S. Senate race in Minnesota and a school board race for the Sausalito Marin City School District in California, a small contest in which voters could vote for up to three candidates.

Sets of Half-Average Nulls Generate Risk-Limiting Audits: SHANGRLA

Philip B. Stark(✉)

University of California, Berkeley, CA, USA
stark@stat.berkeley.edu

Abstract. Risk-limiting audits (RLAs) for many social choice functions can be reduced to testing sets of null hypotheses of the form "the average of this list is not greater than 1/2" for a collection of finite lists of nonnegative numbers. Such social choice functions include majority, super-majority, plurality, multi-winner plurality, Instant Runoff Voting (IRV), Borda count, approval voting, and STAR-Voting, among others. The audit stops without a full hand count iff all the null hypotheses are rejected. The nulls can be tested in many ways. Ballot polling is particularly simple; two new ballot-polling risk-measuring functions for sampling without replacement are given. Ballot-level comparison audits transform each null into an equivalent assertion that the mean of re-scaled tabulation errors is not greater than 1/2. In turn, that null can then be tested using the same statistical methods used for ballot polling—applied to different finite lists of nonnegative numbers. The SHANGRLA approach thus reduces auditing different social choice functions and different audit methods to the same simple statistical problem. Moreover, SHANGRLA comparison audits are more efficient than previous comparison audits for two reasons: (i) for most social choice functions, the conditions tested are both necessary and sufficient for the reported outcome to be correct, while previous methods tested conditions that were sufficient but not necessary, and (ii) the tests avoid a conservative approximation. The SHANGRLA abstraction simplifies stratified audits, including audits that combine ballot polling with ballot-level comparisons, producing sharper audits than the "SUITE" approach. SHANGRLA works with the "phantoms to evil zombies" strategy to treat missing ballot cards and missing or redacted cast vote records. That also facilitates sampling from "ballot-style manifests," which can dramatically improve efficiency when the audited contests do not appear on every ballot card. Open-source software implementing SHANGRLA ballot-level comparison audits is available. SHANGRLA was tested in a process pilot audit of an instant-runoff contest in San Francisco, CA, in November, 2019.

Keywords: Sequential tests · Martingales · Kolmogorov's inequality

1 Introduction

A *risk-limiting audit* (RLA) of a reported election contest outcome is any procedure that guarantees a minimum probability of correcting the reported outcome

Some of Ron's work on affirmative evidence

ClipAudit: A Simple Risk-Limiting Post-Election Audit

Ronald L. Rivest
MIT CSAIL
rivest@mit.edu

January 31, 2017

Abstract

We propose a simple risk-limiting audit for elections, ClipAudit. To determine whether candidate A (the reported winner) actually beat candidate B in a plurality election, ClipAudit draws ballots at random, without replacement, until either all cast ballots have been drawn, or until

$$a - b \geq \beta\sqrt{a+b}$$

where a is the number of ballots in the sample for the reported winner A, and b is the number of ballots in the sample for opponent B, and where β is a constant determined a priori as a function of the number n of ballots cast and the risk-limit α.

ClipAudit doesn't depend on the unofficial margin (as does Bravo).

We show how to extend ClipAudit to contests with multiple winners or losers, or to multiple contests.

Keywords: elections, auditing, post-election audits, risk-limiting audit.

DiffSum – A Simple Post-Election Risk-Limiting Audit

Ronald L. Rivest
MIT CSAIL, rivest@mit.edu

May 10, 2018

We present DiffSum, a simple risk-limiting post-election ballot-polling audit. See [3, 2, 1] for background.

You wish to check that candidate A really won a plurality election against candidate B. You may sample the n cast paper ballots without replacement.

Procedure DiffSum:

1. **[Choose** c**]** Let d be the number of decimal digits in n, and choose $c = d + \delta$ where δ controls the error rate (the chance of the audit accepting an incorrect outcome):

δ	0	1	2	3	4
max error rate	22%	15%	10%	6%	4%

2. **[Begin]** Draw an initial sample of 24 ballots.
3. **[Tally]** Determine the number a of votes for A in your sample, and the number b of votes for B.
4. **[Stop?]** Stop the audit (accept A as winner) if $a > b$ and
$$(a-b)^2 > c \cdot (a+b)\ . \qquad (1)$$
5. **[Continue?]** If $a + b = n$, stop (you have just completed a full recount). Otherwise, enlarge your random sample and return to step 3.

Remarks: The initial size 24 of the sample in step 2 is arbitrary. In step 5 the increase in sample size is also arbitrary; it could be by a single ballot.

The name "DiffSum" was chosen because (1) says

$$(\text{difference})^2 > c \cdot (\text{sum})\ . \qquad (2)$$

Error rate: The error rate bounds given in Step 1 are based on extensive simulations for $\delta = 0$ to 4, $d = 3$ to 7, $n = 10^d$, and $c = d+\delta$. We measured the error rate over 10,000 simulated elections in each case. Each simulation estimated the error rate when the election was a tie, a worst-case scenario; with more realistic margins the error rate drops dramatically, so that in practice even $c = d$ should give very reliable audits.

Example: An election with $n = 50,000$ votes can be audited using $c = 7$ for a risk limit of $\alpha = 10\%$. For $m = 0.20$, DiffSum examines about 175 ballots (estimated), Bravo (with $\alpha = 0.10$) examines about 115 (estimated). In simulations for this election, DiffSum with $c = 7$ examines about 157 ballots on average, and has an error rate of less than 0.04%; DiffSum with $c = 5$ examines about 112 ballots on average, and has an error rate of less than 0.2%. Bravo examines about 119 ballots on average, and has an error rate of approximately 2.5%.

Extension: In practice, one should cease random sampling once a significant number (say 4%) of the ballots have been sampled, when switching over to a full hand recount becomes more economical.

With more candidates, let DiffSum check that the sample winner beats the sample's strongest loser.

Conclusion: DiffSum is exceptionally simple, and appears quite comparable to Bravo in terms of efficiency and error rate. Further simulations and analysis would be helpful.

Acknowledgment: I thank Philip Stark for helpful comments.

References

A Bayesian Method for Auditing Elections

Ronald L. Rivest
Computer Science and Artificial Intelligence Lab,
MIT, Cambridge, MA 02139
rivest@mit.edu

Emily Shen
Computer Science and Artificial Intelligence Lab,
MIT, Cambridge, MA 02139
eshen@csail.mit.edu

Abstract

We propose an approach to post-election auditing based on Bayesian principles, and give experimental evidence for its efficiency and effectiveness. We call such an audit a "Bayes audit". It aims to control the probability of miscertification (certifying a wrong election outcome). The miscertification probability is computed using a Bayesian model based on information gathered by the audit so far.

A Bayes audit is a single-ballot audit method applicable to any voting system (e.g. plurality, approval, IRV, Borda, Schulze, etc.) as long as the number of ballot types is not too large. The method requires only the ability to randomly sample single ballots and the ability to compute the election outcome for a profile of ballots. A Bayes audit does not require the computation of a "margin of victory" in order to get started.

Bayes audits are applicable both to ballot-polling audits, which work just from the paper ballots, and to comparison audits, which work by comparing the paper ballots to their electronic representations. The procedure is quite simple and can be described on a single page.

The Bayes audit uses an efficient method (which may be based on the use of gamma variates or on Pólya's Urn) for simulating a Bayesian posterior distribution on the tally of a profile of ballots.

1 Introduction

This section provides a quick introduction to post-election audits and our notation. Section 2 then presents our proposed Bayes audit procedure. Section 3 gives the results of our initial experiments using this method on simulated and real election data. Section 4 considers some extensions and variations of the basic method, and Sections 5 and 6 discuss and summarize what we have learned about the Bayes audit. Appendix A provides some additional technical details on efficient implementation methods.

1.1 Post-election audits

Informally, the purpose of a post-election audit is to check that the reported election outcome is correct, by auditing enough randomly chosen ballots.

Absolute certainty isn't required of an audit (the only way to achieve absolute certainty is to audit by hand all, or nearly all, of the ballots), but a good audit should have a high probability of exposing (and correcting) an incorrect reported outcome.

The number of ballots audited is typically variable, depending on factors such as the margin of victory (close elections require more work), the random sampling process, whether the audit is a ballot-polling audit or a comparison audit, and (if a comparison audit) the number and nature of errors found. The audit may proceed in stages, auditing more and more ballots until an audit result can

Bayesian Tabulation Audits Explained and Extended

Ronald L. Rivest
MIT CSAIL
rivest@mit.edu

Version February 13, 2018

Abstract

Tabulation audits for an election provide statistical evidence that a reported contest outcome is "correct" (meaning that the tabulation of votes was properly performed), or else the tabulation audit determines the correct outcome.

Stark [51] proposed **risk-limiting tabulation audits** for this purpose; such audits are effective and are beginning to be used in practice in Colorado [38] and other states.

We expand the study of election audits based on **Bayesian** methods. Such Bayesian audits use a slightly different approach first introduced by Rivest and Shen in 2012 [44]. (The risk-limiting audits proposed by Stark are "frequentist" rather than Bayesian in character.)

We first provide a simplified presentation of Bayesian tabulation audits. Suppose an election has been run and the tabulation of votes reports a given outcome. A Bayesian tabulation audit begins by drawing a random sample of the votes in that contest, and tallying those votes. It then considers what effect statistical variations of this tally have on the contest outcome. If such variations almost always yield the previously-reported outcome, the audit terminates, accepting the reported outcome. Otherwise the audit is repeated with an enlarged sample.

We highlight the auditing of such multiple-jurisdiction contests where some of the jurisdictions have an electronic cast vote record (CVR) for each cast paper vote, while the others do not. Complex situations such as this may arise naturally when some counties in a state have upgraded to new equipment, while others have not. Bayesian audits are able to handle such situations in a straightforward manner.

We also discuss the benefits and relevant considerations for using Bayesian audits in practice.

Keywords: elections, auditing, post-election audits, risk-limiting audit, tabulation audit, bayesian audit.

Contents

I don't *always* agree with Ron:

Bayesian Audits Are Average But Risk-Limiting Audits are Above Average

Amanda K. Glazer(✉), Jacob V. Spertus, and Philip B. Stark

Department of Statistics, University of California, Berkeley, CA, USA
{amandaglazer,jakespertus,pbstark}@berkeley.edu

Abstract. Post-election audits can provide convincing evidence that election outcomes are correct—that the reported winner(s) really won—by manually inspecting ballots selected at random from a trustworthy paper trail of votes. Risk-limiting audits (RLAs) control the probability that, if the reported outcome is wrong, it is not corrected before the outcome becomes official. RLAs keep this probability below the specified "risk limit." Bayesian audits (BAs) control the probability that the reported outcome is wrong, the "upset probability." The upset probability does not exist unless one invents a prior probability distribution for cast votes. RLAs ensure that if *this* election's reported outcome is wrong, the procedure has a large chance of correcting it. BAs control a *weighted average probability* of correcting wrong outcomes over a hypothetical collection of elections; the weights come from the prior. In general, BAs do not ensure a large chance of correcting the outcome of an election when the reported outcome is wrong. "Nonpartisan" priors, i.e., priors that are invariant under relabeling the candidates, lead to upset probabilities that can be far smaller than the chance of correcting wrong reported outcomes. We demonstrate the difference using simulations based on several real contests.

Keywords: Election integrity · Risk-limiting audits · Bayesian audits

1 Introduction

The 2016 U.S. Presidential election was attacked by Russian hackers, and U.S. intelligence agencies warn that several nation-states are already mounting attacks on the 2020 election [22,29–31]. Almost every U.S. jurisdiction uses computers to count votes; many use computers to record votes. All computerized systems are vulnerable to bugs, misconfiguration, and hacking [26]. Voters, poll workers, and election officials are also bound to make mistakes [15]. Enough error from any source—innocent or malicious—could cause a losing candidate to appear to win.

The reported tallies will almost certainly be off by at least a little. Were the tallies accurate enough to ensure that the reported winner(s) really won—that the *reported outcome* is correct?

Authors listed alphabetically.

R. Krimmer et al. (Eds.): E-Vote-ID 2020, LNCS 12455, pp. 84–94, 2020.
https://doi.org/10.1007/978-3-030-60347-2_6

Facilitating audits

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 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.

people.csail.mit.edu/rivest/sampler.py

```
# Reference implementation code for pseudo-random sampler
# for election audits or other purposes.
# Written by Ronald L. Rivest
# filename: sampler.py
# url: http://people.csail.mit.edu/rivest/sampler.py
sampler_version = "November 14, 2011"
#
# Relevant to document being produced by an ad-hoc working group chaired
# by Prof. Philip Stark (U.C. Berkeley) regarding election auditing.
# Tested using python version 2.6.7.   (see www.python.org)
# (Will not work with Python version 3, e.g. 3.x.y)
# (Note added 2014-09-07: As per a suggestion by Chris Jerdonek, one should
#  consider this proposal as based on the use of  UTF-8 encoding for strings
#  throughout.  This comment resolves some potential ambiguities about how
#  strings are converted to byte sequences before hashing, and the types of
#  strings input by raw_input, etc.  See
#    https://github.com/cjerdonek/rivest-sampler-tests
# for more discussion and test-cases.
# )

"""
This program provides a reference implementation of a recommended procedure
to pick a random sample of a given size from a specified set of integers.

This program is "open source" (MIT License) and may be freely used in
almost any way whatsoever by others. (Details given below)
```

k-Cut: A Simple Approximately-Uniform Method for Sampling Ballots in Post-Election Audits*

2018

Mayuri Sridhar and Ronald L. Rivest

Massachusetts Institute of Technology, Cambridge MA 02139, USA
mayuri@mit.edu
rivest@csail.mit.edu

Abstract. We present an approximate sampling framework and discuss how risk-limiting audits can compensate for these approximations, while maintaining their "risk-limiting" properties. Our framework is general and can compensate for counting mistakes made during audits. Moreover, we present and analyze a simple approximate sampling method, "k-cut", for picking a ballot randomly from a stack, without counting. Our method involves doing k "cuts," each involving moving a random portion of ballots from the top to the bottom of the stack, and then picking the ballot on top. Unlike conventional methods of picking a ballot at random, k-cut does not require identification numbers on the ballots or counting many ballots per draw. We analyze how close the distribution of chosen ballots is to the uniform distribution, and design mitigation procedures. We show that $k = 6$ cuts is enough for a risk-limiting election audit, based on empirical data, which provides a significant increase in sampling efficiency. This method has been used in pilot RLAs in Indiana and is scheduled to be used in Michigan pilot audits in December 2018.

Keywords: sampling · elections · auditing · post-election audits · risk-limiting audit · Bayesian audit.

Consistent Sampling with Replacement

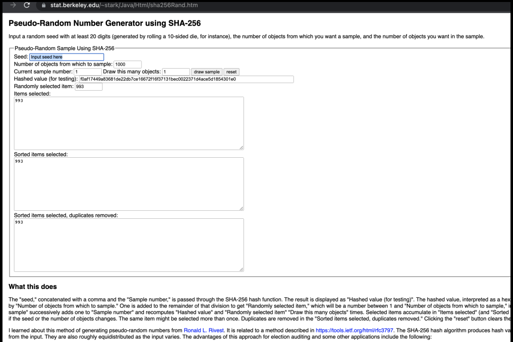
Ronald L. Rivest
MIT CSAIL
rivest@mit.edu

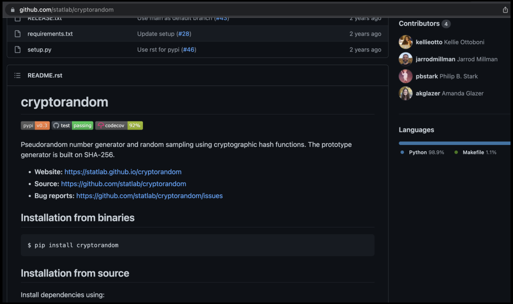
August 31, 2018

Abstract

We describe a very simple method for "consistent sampling" that allows for sampling with replacement. The method extends previous approaches to consistent sampling, which assign a pseudorandom real number to each element, and sample those with the smallest associated numbers. When sampling with replacement, our extension gives the item sampled a new, larger, associated pseudorandom number, and returns it to the pool of items being sampled.

Ripping off Ron: PRNGs





Pedagogy: secrecy and verifiability without cryptography

The ThreeBallot Voting System

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

October 1, 2006*

Abstract

We present a new paper-based voting method with interesting security properties. The attempt here is to see if one can achieve the same security properties of recently proposed cryptographic voting protocols, but without using any cryptography, using only paper ballots. We partially succeed. (Initially, I thought the proposal accomplished this goal, but several readers discovered a vote-buying attack (see Section 4.4) that appears to be rather difficult to fix without making the resulting system much less usable in practice. Currently, this paper should thus be viewed more as an academic proposal than a practical proposal. Perhaps some variation on these ideas in this paper might still turn out to be of practical use. The "OneBallot with Exchanged Receipts" system sketched at the end of Section 5.3.1, looks particularly promising at the moment...)

The principles of ThreeBallot are simple and easy to understand.

In this proposal, not only can each voter verify that her vote is recorded as she intended, but she gets a "receipt" that she can take home that can be used later to verify that her vote is actually included in the final tally. Her receipt, however, does not allow her to prove to anyone else how she voted.

In this "ThreeBallot" voting system, each voter casts *three* paper ballots, with certain restrictions on how they may be filled out, so the tallying works. These paper ballots are of course "voter-verifiable." All ballots cast are scanned and published on a web site, so anyone may matching her receipt. Deletion or modification of ballots is thus detectable; so the integrity of the election is verifiable.

1 Introduction

Designing secure voting systems is tough, since the constraints are apparently contradictory. In particular, the requirement for voter privacy (no one should know how Alice voted, even if Alice wants them to know) seems to contradict verifiability (how can Alice verify that her vote was counted as she intended?).

The proposal presented here is an attempt to satisfy these constraints *without* the use of cryptograpy. We get pretty close...

Like most cryptographic proposals, ThreeBallot uses a public "bulletin board"–a public web site where election officials post copies of all of the cast ballots (there will be $3n$ of them if there are n voters) and a list of the names of the voters who voted. (Some states might use voter ID's rather than voter names.)

One key principle of ThreeBallot is to "vote by rows" and "cast by columns". The ThreeBallot ballot can viewed as an array, where the voter places marks in rows corresponding to candidates, but then separates the columns and casts them separately, keeping a copy of one.

ThreeBallot provides a nice level of end-to-end verifiability—the voter gets assurance that her vote was cast as intended and counted as cast, and that election

---

- You have here three optical scan ballots arranged as three columns; you will be casting all three ballots.
- Proceed row by row through the multi-ballot. Each row corresponds to one candidate. There are three "bubbles" in a row, one on each ballot.
- To vote **FOR** a candidate, you must fill in exactly two of the bubbles on that candidate's row. You may choose arbitrarily which two bubbles in that row to fill in. (It doesn't matter, as all three ballots will be cast.)
- To vote **AGAINST** a candidate (i.e., to not vote **FOR** the candidate, or to cast a "null" vote for that candidate), you must fill in exactly one of the bubbles on that candidate's row. You may choose arbitrarily which bubble in that row to fill in. (It doesn't matter, as all three ballots will be cast.)
- You *must* fill in *at least one* bubble in each row; your multi-ballot will not be accepted if a row is left entirely blank.
- You *may not* fill in *all three* bubbles in a row; your multi-ballot will not be accepted if a row has all three bubbles filled in.
- You may vote **FOR** at most one candidate per race, unless indicated otherwise (In some races, you are allowed to vote **FOR** several candidates, up to a specified maximum number.) It is OK to vote **AGAINST** all candidates.

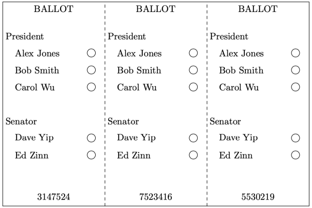


Figure 1: A sample ThreeBallot multi-ballot, with a first race for President with candidates Jones, Smith, and Wu and a second race for Senator with candidates Yip and Zinn.

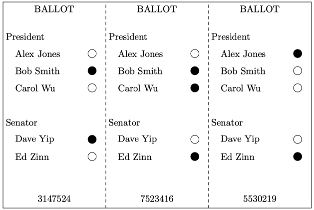
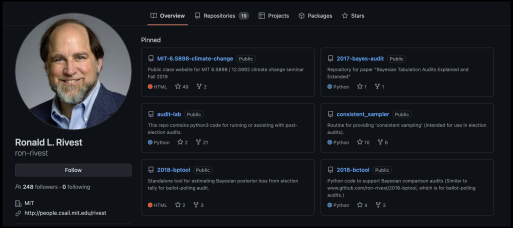
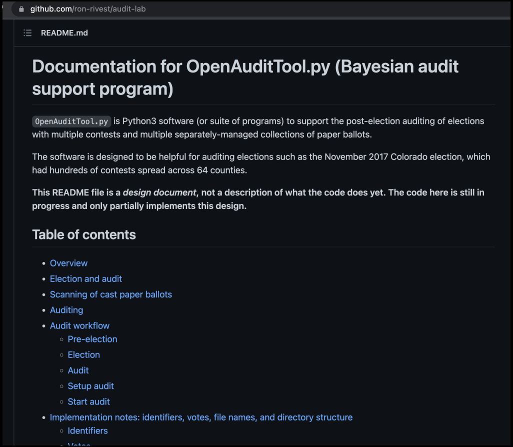
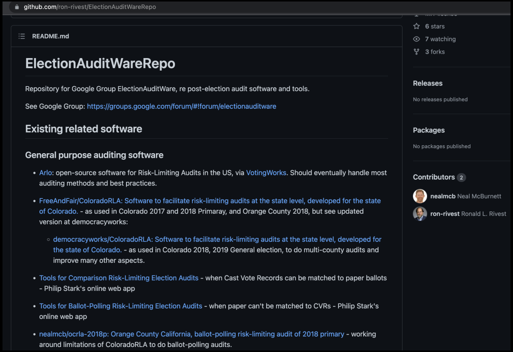


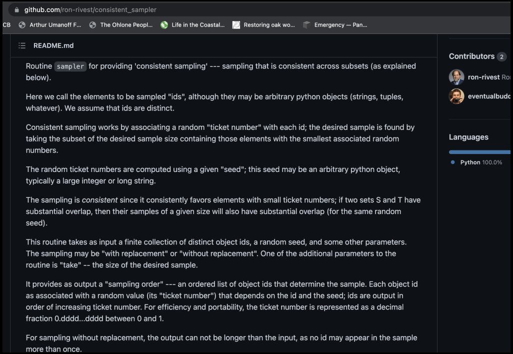
Figure 2: A filled-out version the multi-ballot of Figure 1, showing a vote **FOR** Smith for President and a vote **FOR** Zinn as Senator, since the rows for these candidates have two filled-in bubbles (marks) each. All other rows have exactly one mark. (There are many other ways such choices could have been indicated.) Note that ballot 7523416, when viewed as a conventional ballot, looks like an overvote for President.

Open-source software






github.com/ron-rivest/ElectionAuditWareRepo
README.md
ElectionAuditWareRepo
Repository for Google Group ElectionAuditWare, re post-election audit software and tools.
See Google Group: https://groups.google.com/forum/#!forum/electionauditware
Existing related software
General purpose auditing software
Arlo: open-source software for Risk-Limiting Audits in the US, via VotingWorks. Should eventually handle most auditing methods and best practices.
FreeAndFair/ColoradoRLA: Software to facilitate risk-limiting audits at the state level, developed for the state of Colorado. - as used in Colorado 2017 and 2018 Primaray, and Orange County 2018, but see updated version at democracyworks:
democracyworks/ColoradoRLA: Software to facilitate risk-limiting audits at the state level, developed for the state of Colorado. - as used in Colorado 2018, 2019 General election, to do multi-county audits and improve many other aspects.
Tools for Comparison Risk-Limiting Election Audits - when Cast Vote Records can be matched to paper ballots - Philip Stark's online web app
Tools for Ballot-Polling Risk-Limiting Election Audits - when paper can't be matched to CVRs - Philip Stark's online web app
nealmcb/ocrla-2018p: Orange County California, ballot-polling risk-limiting audit of 2018 primary - working around limitations of ColoradoRLA to do ballot-polling audits.
6 stars
7 watching
3 forks
Releases
No releases published
Packages
No packages published
Contributors 2
nealmcb Neal McBurnett
ron-rivest Ronald L. Rivest


github.com/ron-rivest/consistent_sampler
Arthur Umanoff F...
The Ohlone Peopl...
Life in the Coastal...
Restoring oak wo...
Emergency — Pan...
README.md
Routine sampler for providing 'consistent sampling' --- sampling that is consistent across subsets (as explained below).
Here we call the elements to be sampled "ids", although they may be arbitrary python objects (strings, tuples, whatever). We assume that ids are distinct.
Consistent sampling works by associating a random "ticket number" with each id; the desired sample is found by taking the subset of the desired sample size containing those elements with the smallest associated random numbers.
The random ticket numbers are computed using a given "seed"; this seed may be an arbitrary python object, typically a large integer or long string.
The sampling is consistent since it consistently favors elements with small ticket numbers; if two sets S and T have substantial overlap, then their samples of a given size will also have substantial overlap (for the same random seed).
This routine takes as input a finite collection of distinct object ids, a random seed, and some other parameters. The sampling may be "with replacement" or "without replacement". One of the additional parameters to the routine is "take" -- the size of the desired sample.
It provides as output a "sampling order" --- an ordered list of object ids that determine the sample. Each object id as associated with a random value (its "ticket number") that depends on the id and the seed; ids are output in order of increasing ticket number. For efficiency and portability, the ticket number is represented as a decimal fraction 0.dddd...dddd between 0 and 1.
For sampling without replacement, the output can not be longer than the input, as no id may appear in the sample more than once.
Contributors 2
ron-rivest
eventualbudd
Languages
Python 100.0%

Influence on my work:

- taught me more than I remember about mixnets, ZKP, homomorphic encryption, E2E-V, distributed ledgers
- detection audits led to risk-limiting audits
- software independence led to evidence-based elections, contestability, defensibility & formalizing SI
- improving RLAs
- better PRNGs for statistics and audits


Napa County, CA, 2012