Scan, Shuffle, Rescan: Machine-Assisted Election Audits With Untrusted Scanners

Douglas W. Jones
University of Iowa

Sunoo Park
Cornell Tech

Ronald L. Rivest
MIT CSAIL

Adam Sealfon
UC Berkeley

Abstract

We introduce a new way to conduct post-election audits using untrusted scanners. Audits perform statistical hypothesis testing to confirm election outcomes. However, existing approaches are costly and laborious for close elections—often the most important cases to audit—requiring extensive hand inspection of ballots. We instead employ automated consistency checks, augmented by manual checks of only a small number of ballots. Our protocols scan each ballot twice, shuffling the ballots between the scans. This gives strong statistical guarantees even for close elections, as long as (1) the permutation accomplished by the shuffle is unknown to the scanners and (2) the scanners cannot reliably identify a particular ballot among others cast for the same candidate. In practice, ballots often have distinguishing features, of course; but we argue that reasonable measures can limit their detection by scanners under controlled conditions. Our techniques drastically reduce the time, expense, and labor of auditing close elections, which we hope will facilitate wider deployment.

We present three rescan audit protocols and analyze their statistical guarantees. We first present a simple scheme illustrating our basic idea in a simplified two-candidate setting. We then extend this scheme to allow (1) more than two candidates; (2) processing of ballots in batches; and (3) tolerating imperfect scanners, as long as scanning errors are too infrequent to affect the election outcome. Finally, we propose and discuss an alternate scheme that reduces the trust assumptions placed on the shuffling mechanism at the expense of adding an additional scan. Our proposals require manual handling or inspection of 10–100 ballots per batch in a variety of settings, in contrast to existing techniques that require hand inspecting many more ballots in close elections. Unlike prior techniques that depend on the *relative* margin of victory, our protocols are to our knowledge the first to depend on the *absolute* margin, and give meaningful guarantees even for extremely close elections: e.g., absolute margins of tens or hundreds of votes.

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

1 Introduction and motivation

Election outcomes are determined by tabulating the votes cast in the election and identifying the winner: for plurality elections, the winner is the candidate who received the most votes. In the United States, the electorate is relatively large and ballots are often complex, and (unusually) ballot processing and tabulation are typically performed by machine [28].

Machines provide efficiency, but do not guarantee accuracy. Individuals, corporations, and nation-state actors all have strong incentives to influence the results of political elections. Even absent deliberate tampering, election machinery—for scanning, tabulation, or otherwise—may have software bugs or be misconfigured for a particular election. Such factors can cause incorrect election outcomes that may be hard to detect.

Post-election audits [6, 25] provide safeguards to assure election officials and the public that the ballots cast were tabulated and reported accurately—or alert them if not. The standard way to conduct a post-election audit is to (1) inspect a random sample of ballots by hand, and (2) assess the likelihood of such a sample supposing the reported outcome was incorrect. This can give a rigorous statistical guarantee about the election outcome's likely correctness based on the sample, without the need to hand-count every ballot. But such statistical guarantees come at a cost that can be significant to often underresourced local election officials [28]. In close elections, statistical audits require laborious manual inspection of many ballots, and in very close elections a full hand recount may be needed to get a meaningful statistical guarantee (such as in Georgia's 2020 election in the U.S. [18]).

Recognizing the importance of verifying election results and detecting errors, some states now require post-election audits by law for at least some contests, and all states allow recounts for close elections [28, 38]. Recent U.S. elections and political discourse (e.g., [24, 29, 41]) further underscore the need for transparency and public confidence in electoral systems. Such confidence is as much a sociopolitical as a technical phenomenon: as such, technical transparency and verifiability are needed, but not sufficient.

In this paper, we ask: can *partial automation* improve post-election audit efficiency for *close elections*, by reducing the costly manual labor required? In answer, we propose a new kind of post-election audit, called a *rescan audit*, with the potential to reduce labor in close elections to handling just tens of ballots for a range of realistic parameters—at the cost of 2–3 times overhead in mechanical ballot processing, and two assumptions on communication and ballots (Section 3.1). The overhead and assumptions are more appropriate and plausible in certain election contexts than others, and the assumptions are *not* plausible for all election contexts, as we detail later.

Our rescan audits compare the ballot-by-ballot results from two separate scans of all ballots. The second scan provides consistency checks that can be used to obtain statistical guarantees for the correctness of the election outcome reported by the first scan, without trusting either scanner to behave correctly. In practice, an additional scan of the ballots is already sometimes performed, for auditing or other purposes [7, 8, 19].

However, a second scan alone is insufficient to guarantee election integrity in the presence of colluding adversarial scanners. For example, the scanners may agree in advance on a set of positions and misreport the votes on ballots in those positions. If both scanners operate on the same sequence of ballots, their outputs would appear consistent. Similarly, if the scanners are misconfigured the same way—e.g., if they ignore the first batch of ballots, or are preloaded with the results of a prior election—they will produce consistent incorrect outputs.

Thus, an additional scan only offers a useful guarantee if the scanners cannot coordinate their misreporting. To prevent coordination, we shuffle the ballots between scans so the scanners do not observe the ballots in the same order. Thus, adversarial scanners will be unable to consistently misreport the same set of ballots—unless they misreport *all* ballots cast for some candidate. We can detect such extreme misbehavior by manually inspecting just a few ballots. Our audits are built from this basic sequence: *scan, shuffle, rescan*, supplemented with manual handling or inspection of only a few ballots.

Figure 1 gives an overview of our rescan audit workflow. The set of ballots $\mathbf{x}$ is scanned on scanner S_1, to give a sequence of cast-vote records (CVRs) $\mathbf{y}$ indicating how the scanner interpreted each ballot. A labeler L then applies random-looking unique identifiers (labels) to the ballots, after which the ballots are permuted by shuffler Π. The ballots are then scanned on a second scanner S_2 (possibly the same as S_1), yielding a second list $\mathbf{z}$ of CVRs. Because each scan processed the same ballots, every CVR in $\mathbf{y}$ should also appear in $\mathbf{z}$, but the two scanners see the ballots in seemingly unrelated orders (so the order of CVRs in $\mathbf{y}$ differs from that in $\mathbf{z}$). Hence, erroneous or adversarial scanners would have an extremely low chance of misreporting exactly the same ballots in $\mathbf{y}$ and $\mathbf{z}$.

The comparison logic does ballot-level comparison, finding corresponding CVRs in $\mathbf{y}$ and $\mathbf{z}$. To do this, it must know how the collection of ballots was permuted. This is achieved using

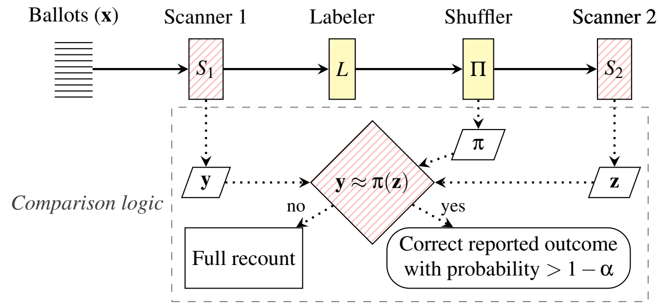

Figure 1: Flow diagram of an election with a rescan audit. Pink (hatched) components are untrusted (i.e., may have been corrupted by an adversary); yellow (solid) components are trusted, and unable to examine the votes on the ballots. See Section 3.1 for more detail on the threat model.

the labels applied before the shuffle, which can be read by S_2 and included in the CVRs in $\mathbf{z}$. The labels can be generated using a secret key shared by the labeler and the comparison logic, but unknown to S_2. This allows the comparison logic to reconstruct the order of ballots seen by S_1, while ensuring that S_2 cannot do so.[1]

In addition to comparing the CVRs in $\mathbf{y}$ and $\mathbf{z}$, our protocols require manual inspection of a small number of ballots. For single-batch two-candidate elections, a hand inspection of a few ballots sampled at random is sufficient. In more general settings, we introduce *test ballots* for each candidate that are distinguishable from real ballots by humans but not by the scanners (e.g. by edge markings). Test ballots allow us to ensure that *all* candidates, including those who received only a few votes, were correctly allocated their votes in each batch.

Labeling of ballots is not new. It is well established practice in ballot-comparison audits to add labels to voted ballots so that the electronic records scanned from those ballots can be matched to the original paper ballots [27].

Evaluation We conduct an evaluation based on timing and cost data from the Rhode Island pilot study of risk-limiting audits [19]: while audit costs are likely to vary significantly, the RI report provides the best documentation currently publicly available on the costs and timings of risk-limiting audits under realistic conditions. Our analysis shows that for elections with margins under roughly 1%, rescan audits are competitive with or better than the best known statistical risk-limiting audits. The metrics we use for evaluation are: (1) number of ballots that must be handled manually, based on our security proofs; (2) estimated timings, based on the timings of key audit operations as documented in the Rhode Island pilot study and subsequent research; and (3) labor costs (excluding training and equipment), again based on Rhode Island data.

[1]Alternatively, in place of the labeler, one could use a keyed shuffle to reorder the ballots in a way known to the comparison logic but unknown to S_2. Keyed shuffles pose more practical difficulty than unkeyed shuffles, as further discussed in Section 7.

Risk-limiting auditing systems [25] typically require hand inspection of a number of ballots dependent on the *proportional* margin. To our knowledge, our protocols are the first where the efficiency of the manual audit depends instead on the *absolute* (reported) margin. We achieve this due to the fact that our rescan procedure already induces consistency checks on all ballots without any manual examination whatsoever, whereas the size of the sample of ballots inspected by a traditional hand audit must depend on the relative margin.

Since manual labor in rescan audits depends only weakly on the margin, the workload is *identical* for a wide range of margins.[2] This means rescan audit workload is highly predictable in advance. In contrast, traditional risk-limiting audit workload varies much more, depending on the initial sample of ballots: particularly for close elections, unlucky samples may lead to workload escalation even to a full recount. The greater predictability of rescan audits is desirable for real-world audits subject to budgetary constraints and statutory deadlines.

Conventional risk-limiting audits can be more efficient for elections with wide margins, so in practice, a rescan audit could be invoked only when an election is close. Alternatively, rescan audits may be a desirable alternative when a conventional risk-limiting audit requires a full hand recount.

Summary of contributions We present three rescan audit protocols, as summarized in Table 1. We provide security proofs and an evaluation for two protocols, and give a preliminary analysis and security conjecture for the third.

- We introduce a new paradigm, *rescan audits*, which utilizes additional ballot scans to substantially reduce manual labor in statistical post-election audits, subject to clearly specified assumptions.
- BASICAUDIT (§4) is the simplest and most stylized protocol, adequate for a single two-candidate contest where all ballots are perfectly readable by the scanners and have no distinguishing features (beyond the vote itself).
- 2SCANAUDIT (§5) accounts for *multiple-candidate* contests, where auditing may be done in *batches* (e.g., by precinct), and supports *imperfect scanners* that may make errors. 2SCANAUDIT, unlike BASICAUDIT, involves mingling clearly marked *test ballots* with the voted ballots.
- 3SCANAUDIT (§6) explores an approach to reducing the trust assumptions of the first two protocols, at the expense of a third scan. While the other protocols require trust in both the programmable labeler and the shuffler, 3SCANAUDIT relies only on a purely mechanical shuffler. Unlike the other protocols, 3SCANAUDIT lacks a formal security proof: as such, it is a more exploratory direction.
- Our rigorous ***security proofs*** for BASICAUDIT and 2SCANAUDIT (§§B–C) allow precise specification of parameter tradeoffs for different risk limits.
- Our ***evaluation*** (§8) indicates that for close elections, rescan audits are competitive with or better than the best known statistical risk-limiting audits, in terms of number of ballots handled manually, timing, and labor cost.

[2] E.g., for absolute margins of at least 100 votes, with risk limit $\alpha = 0.05$.

		BASIC	2SCAN	3SCAN
# candidates	**2**	✓	✓	✓
	> 2		✓	✓
Batch audits			✓	✓
Robust to scanner errors			✓	✓
No trusted shuffle				✓
Provable security		✓	✓	**?**

Table 1: Summary of rescan audit protocols

Prior work Introductions to post-election audits can be found in Stark [32] (which introduces the notion of *risk-limiting tabulation audits*), Stark and Wagner [31], Lindeman and Stark [25], Bretschneider et al. [6], and Verified Voting [39].

Calandrino et al. suggested using scanners to assist in ballot-level audits, in 2007 [8]. They used an auditing tabulator that labels ballots with identifying numbers so that high assurance could be achieved by hand checking the scans of a few randomly selected ballots. We improve on this with a second scan of all ballots instead of manual checking of a few. Rescanning itself is not new: e.g., "transitive audits" [25] use a rescan to process ballot labels. Rescanning was pioneered in Humbolt County, California [15], and the Clear Ballot Group's *ClearAudit* was certified in Florida in 2014 [17]. These audits rely on independently developed scanners and tabulation software to impede collusion between the scans. Our proposal is the first to suggest rescanning for comparison between scanned ballot values.

The developers of the *Rijnland Internet Election System* found that mingling test ballots with real ballots provides a useful test of voting system integrity [23]. Our use of test ballots is different, but we also mix them with real ballots.

Finally, our ideas are inspired by *multi-prover proofs* in cryptography [3], but our techniques differ because of the concrete statistical guarantees we seek and the practical constraints of physical ballot processing. Moreover, two-prover proofs require that the provers be unable to communicate, while our non-communication assumption is weaker, allowing the two scanners to communicate.

2 Background and terminology

An ***election*** has one or many ***contests***. A general election may have many contests, each between a number of ***candidates***. We focus initially on auditing a single two-candidate contest. We assume that all contests are ***plurality*** contests.

A ***voter*** submits (***casts***) a paper ***ballot*** that indicates the voter's ***selections*** (or ***votes***) for each contest. We assume that each ballot is a single piece of paper, ignoring elections with multi-page ballots. A paper ballot is ***voter-verifiable***: a voter can confirm before casting that it correctly records his/her votes. A human examining the voter's cast paper ballot will see the voter's ***correct*** or ***true*** selection for each contest.

Ballots may be scanned individually when cast, or may be collected into ***batches*** for later scanning—say, one per precinct. Ballots may be labeled with a unique ***ballot label*** by a ***labeler***.[3] The ballot label, if visible at the time of scanning, is included in the CVR for the ballot.

Test ballots are ballots that the humans, but not scanners, can distinguish from ***cast ballots***, those from real voters. Test ballots are mingled with the cast ballots before the first scan to create a stream of ***test-or-cast ballots***. They remain mixed with the ballots until the end, when they can be separated by hand from the cast ballots. Because the votes on the test ballots are known in advance, the votes can be subtracted from the totals regardless of the disposition of the test ballots.[4]

The ***reported contest tally*** says how many votes are reported for each candidate in the contest. The ***reported contest outcome*** or ***reported winner*** is the candidate with the most reported votes. The ***true winner*** is defined similarly, based on the true votes cast in that contest. The ***reported margin of victory*** is the difference between the numbers of votes reported for the reported winner and the reported runner-up.

2.1 Scanners

An ***optical scanner*** (or ***scanner*** for short) reads a sequence of cast paper ballots to produce a ***cast vote record*** (or ***CVR***) for each ballot scanned. The CVR is an electronic record giving the scanner's interpretation of the voter's selection for each contest, known as the ***reported*** selection. If a ballot has been labeled, we ask the scanner to include the ballot label in the CVR. Scanners may also provide digital images of each ballot scanned. We allow scanners to examine all ballots in a batch before producing the file of CVRs for those ballots.

Scanners are much faster than hand counting. While precinct-count scanners are relatively slow, central-count scanners able to scan 800 to 1000 ballots per minute have been made [35] and speeds of 300 ballots per minute are now common [2, 30]. Many high-speed scanners can be programmed to deliver ballots into 2 or 3 output hoppers to help separate ballots requiring human interpretation from others (required by the 1990 FEC Voting System Standards [13, §3.2.5.1.1]).

For a ***perfect*** scanner, reported votes equal true votes except when it deliberately cheats. Real scanners are not perfect; errors may occur for many reasons. The Help America Vote Act [20, §301(a)(5)] refers to the FEC 2002 Voting System Standards to require that voting systems have an error rate of "no more than one in 10,000,000 ballot positions;" this excludes human factor issues such as how voters respond to voting instructions [14, §3.2.1]. A re-scan audit of 60,000 voting target images from a 2008 California primary showed that 0.25% were marginal [15]. In the 2008 Minnesota Senate recount, 0.01% of the ballots were marked in a way that even humans could not interpret, and 0.09% of the ballots were classified differently by people than by machine [1].

The ability to sort out ballots requiring interpretation was introduced for handling overvotes and write-in votes. Early scanners had one threshold, so marks in each voting position were classified as either present or absent. A ***marginal mark*** (one near the scanner threshold) might be classified as a vote on one scanning pass and not as a vote on another pass. More recently, EAC Voluntary Voting System Guidelines required two thresholds, where marks between them are classified as marginal and flagged for human attention [12, §1.1.6]. Some scanners do this by diverting such ballots into a separate output stack [30] while others use ***online adjudication***, displaying the problematic mark to human adjudicators and recording their decision in the CVR [22].

The ES&S patent for the DS850 scanner discusses the possible uses of a printer on the scanner's paper path (referred to as an ink cartridge) [2, 30]. One of the authors has observed the use of this scanner to label ballots in a risk-limiting audit.

2.2 Audits

A ***manual post-election audit*** may be used to assure that reported contest outcomes are correct when imperfect scanners are used. In these, cast ballots are selected for hand examination. A ***manual recount*** examines and tallies *all* cast ballots by hand, while a ***statistical audit*** examines only a random sample. When ballots are tabulated in batches, a ***batch-level*** audit may be used, recounting randomly selected batches to check batch tallies. All manual audits require auditors able to hand-interpret ballots.

Statistical audits are very efficient when the fractional margin in an election is large. These come in two main types: In ***ballot-comparison audits***, humans examine ballots in the sample to compare each with the corresponding CVR, while in ***ballot-polling audits***, humans count the votes in the sample to see if the manual count provides statistical support for the reported election outcome. A ***risk-limiting audit*** examines an increasingly larger sample of cast ballots in such a way that the total chance of stopping the audit and accepting an *incorrect* reported outcome is bounded by a given ***risk limit***. Risk limits used by U.S. jurisdictions in practice range between 1–10% [11, 19]. All statistical audits require a process for drawing random samples [33].

[3] We assume throughout that ballot labels are unique. This is without loss of generality: while an adversarial labeler could print non-unique labels, this would cause a detectable discrepancy and so would only harm the adversary.

[4] As a further fail-safe mode — in case of extensive controversy over an election such that such subtraction is insufficient to restore public confidence — removing all the test ballots can be achieved with comparable work to a full recount. The human-visible distinguishability of test ballots means that public observation of the recount-and-remove-test-ballots process would provide credible public assurance of the test ballots' removal.

3 Model

Basic notation We use boldface (e.g., $\mathbf{v}$) to denote vectors, and subscripts to denote elements of a vector (e.g., v_i is the ith element of $\mathbf{v}$). For $N \in \mathbb{N}$, $[N] = \{1, \ldots, N\}$. We write Sym_N to denote the set of all permutations of N elements. If $\mathbf{v}$ is an N-element vector and $\pi \in \mathrm{Sym}_N$, we write $\pi(\mathbf{v})$ to denote the result of applying permutation π to the positions of the elements of $\mathbf{v}$ (i.e., "shuffling" the elements of $\mathbf{v}$). For $i \in [N]$, we write $\pi(i)$ to denote the index to which π maps its ith input element. For sets X, Y, $f : X \to Y$ means f is a function that maps each element of X to an element of Y. For any vector $\mathbf{x}$, the number of nonzero elements of $\mathbf{x}$ is denoted by $\|\mathbf{x}\|_0$. For $c \in [C]$, let $c^{(t)}$ denote the t-tuple $(c, \ldots, c)$.

Ballots and ballot types Let N denote the number of ballots cast in an election; we assume N is known to all parties and devices. We assume that the election is for a single contest, unless stated otherwise. For a given contest, let C denote the number of candidates in the contest, and let the list of ballots cast for that contest be denoted by a vector $\mathbf{x}$ in $[C]^N$. We use M to denote an absolute margin of victory and $m = M/N$ to denote a relative margin.

The ***type*** of a ballot is defined by the candidate preference indicated on the ballot: in a C-candidate contest, there are C ballot types (for simplicity of modeling, we do not consider undervotes as a separate type). For $c \in [C]$, if a ballot indicates a preference for candidate c, we call it a ballot of type c.

Manual ballot inspection $\textsc{Recount}(\mathbf{x})$ denotes a full manual recount of the ballots $\mathbf{x}$; it outputs a vector of results $\mathbf{y}$ (and thus, implicitly, the true election outcome as well; the manual recount is correct by definition). $\textsc{HandInspect}_h(\mathbf{x})$ denotes a hand inspection of the first h randomly sampled ballots among $\mathbf{x}$; we will apply it to shuffled piles of ballots so that it represents a manual ("spot" or "hand") check of random ballots. It outputs a vote vector $\mathbf{v} = (v_1, \ldots, v_h)$ consisting of the results of the hand inspection of these ballots. Discrepancies found in such a manual check (with respect to reported values for those ballots) are called ***manual discrepancies***.

Hardware components Our audit procedures use three types of hardware components that handle paper ballots: scanners, labelers, and shufflers (as shown in Figure 1). ***Scanners*** are extensively discussed in Section 2.1. A ***labeler*** is a machine that takes a set of ballots and prints numbers or strings onto a specified part of each ballot (e.g., the left edge).

A ***shuffler*** is a machine or procedure that takes a set of ballots in a certain order and permutes them into a different order. The permutation may or may not be known to the shuffler but is assumed to be random.[5] An ***unknown-shuffle*** procedure $\tilde{\mathbf{x}} \leftarrow \Pi(\mathbf{x})$ scrambles the order of ballots in a pile without knowing or revealing the permutation (consider strewing the pile of ballots on the floor and picking them up again, although that would not be appropriate in practice). A ***known-shuffle*** procedure $(\tilde{\mathbf{x}}, \pi) \leftarrow \Pi(\mathbf{x})$ outputs the permutation π alongside the shuffled ballots $\tilde{\mathbf{x}}$ (such that $\tilde{\mathbf{x}} = \pi(\mathbf{x})$).[6] Section 7 discusses practical ways to achieve a known shuffle.

A shuffle procedure Π is ***hiding*** if no efficient adversary given $\tilde{\mathbf{x}}$ can learn any information about π. We require this property. Both known and unknown shuffles can be hiding, since the term *(un)known* refers to whether the shuffler itself knows the permutation, whereas the *hiding* property refers to whether the permutation can be learned just by looking at the output ballot stack.

A ***scanner function*** is a function S that maps a sequence $\mathbf{x}$ of cast votes to a same-length sequence $\mathbf{y}$ of cast vote records. We refer to a scanner as ***misreporting*** the ith ballot for input $\mathbf{x}$ if $\mathbf{x}_i \neq \mathbf{y}_i$, i.e., if its output differs from its input at index i.

Audit terminology A ***risk limit*** $\alpha \in [0, 1]$ is an upper bound on the conditional probability that if there is an error in the reported election outcome that the audit will fail to detect it. In other words, an audit with risk limit α will detect an error in the reported election outcome with probability at least $1 - \alpha$ (subject to any trust assumptions or cryptographic assumptions on which the statistical guarantees of the audit are based).

Our terminology is consistent with the usual definition of a risk limit for a risk-limiting audit: whenever our audit detects an error in the reported outcome, it proceeds to a full recount to determine the correct election outcome, just as a standard risk-limiting audit escalates to a full manual recount.

A ***rescan audit procedure*** $\textsc{Audit}$ for a single contest takes as input a sequence of N ballots $\mathbf{x}$, a risk limit α, and (optionally) some additional parameters, and outputs $(\tau, \mathbf{y})$ where $\tau \in \{\texttt{hand}, \texttt{auto}\}$ and y is a vector of length N giving the results of scanning or manually inspecting the ballots in x. We say $\textsc{Audit}$ ***outputs the correct winner*** if the winning candidate $c_{\mathbf{y}}$ based on the results $\mathbf{y}$ outputted by $\textsc{Audit}$ is equal to the true winner $c_{\mathbf{x}}$ of the contest. If $\tau = \texttt{auto}$, the accompanying $\mathbf{y}$ represents the results of an optical scan. If $\tau = \texttt{hand}$, the accompanying $\mathbf{y}$ represents the results of a full manual recount (which are correct by definition). The rescan audit procedure should be accompanied by provable guarantees that the output $\mathbf{y}$ reflects the correct election outcome (i.e., the correct winner for each contest)[7] with probability at least $1 - \alpha$ (subject to any explicitly stated conditions or assumptions). We say that an audit procedure ***accepts*** if it outputs $(\texttt{auto}, \cdot)$, denoting that it accepts the optical scan results.

3.1 Threat models and assumptions

Our protocols are designed for two different threat models.

[5] Assuming a uniformly random shuffle simplifies the analysis but is not strictly necessary; a sufficiently entropic shuffle would suffice.

[6] A ***chosen-shuffle*** procedure taking the permutation to be implemented as an input is more demanding, but a *known-shuffle* suffices for our purposes.

[7] Our election audits serve to check the *outcome* or *winner*, not the specific tallies for each candidate. In particular, $\mathbf{y}$ may reflect the correct election outcome even if some of its reported ballot types are incorrect.

Threat Model 1 BASICAUDIT *and* 2SCANAUDIT *rely on the following set of trust assumptions (also expressed graphically in Figure 1 in Section 1):*

- *The* ***scanners*** *are* untrusted *(indicated in pink).*
- *The* ***comparison logic****—that is, the software used to compare the scanners' results—is* untrusted *(again pink), since its output can be independently verified.*
- *The* ***labeler*** *and* ***shuffler*** *used to implement a known shuffle are* trusted *not to communicate with each other (indicated in yellow). They* need not be trusted *to correctly implement a particular known shuffle.*[8]

Threat Model 2 3SCANAUDIT *is designed for the following stronger, and thus preferable, threat model (also expressed graphically in Figure 2 in Section 6):*

- *The* ***scanners*** *and* ***comparison logic*** *are* untrusted *(pink).*
- *The* ***shuffler*** *is purely mechanical; its reliable mechanical operation is trusted but it* requires no trusted software *(indicated in blue).*[9] *Again, the shuffler* need not *implement a particular known shuffle.*
- *The* ***printer****, a new component not present in the other protocols, is* untrusted *(pink).*

We treat *untrusted* components as behaving arbitrarily, and possibly colluding with one another. Our protocols guarantee correction or detection of any errors due to adversarial (or otherwise erroneous) behavior of untrusted components. Our formal model and theorems assume that *trusted* components behave correctly. Section 7 discusses mitigating measures that could significantly improve our assumptions on trusted components in practice.

In order to avoid the necessity of trusting the comparison logic, we assume that both scanners' outputs (CVRs and labels) are publicly released. This allows the comparison logic to be verified independently by any observer.

Assumptions We rely on three key assumptions, discussed below. The suitability of our protocols depends on the realism of these assumptions in specific application contexts, as well as the costs and benefits of a rescan audit approach in context. We discuss, here and in Section 7, a number of concrete mitigating measures (in system design, setup, and election administration) to bolster these assumptions' plausibility.

Our approach is not suitable for all election contexts, but we believe that our proposals provide real benefits compared to prior election auditing models when the margin is close, and that the mitigating measures we propose make our protocols useful in a range of realistic settings.

- ***Non-communication assumption.*** We rely on the assumption (implicit in our threat models) that *during the audit*, the *labeler and shuffler* do not communicate with the *second scanner*. Our protocols are secure against arbitrary communication *between the two scanners*, and against arbitrary collusion between the labeler, shuffler, and scanners *before the audit* (e.g., they could be preprogrammed with a coordinated malicious strategy and shared secrets, whether by hardware/software developers or upstream supply chain links). Our non-communication assumption is needed because an adversarial labeler could otherwise transmit the permutation to the second scanner: e.g., by a covert wireless channel or by steganographic encoding in ballot labels. The 2021 EAC Guidelines [12, §14.2.E] strictly limit network connectivity in voting equipment. Non-communication between system components conforming to these guidelines can be enforced by physically separating machines, removing wireless ports, sealing wired ports, and limiting physical access. Note that close observation by officials and outside observers is a standard requirement for elections [9, 16, 36].
- ***Ballot indistinguishability assumption.*** Our results require that the scanners cannot identify a particular ballot among others cast for the same candidate. That is, we assume the ballots do not contain identifying marks that the scanners could use to coordinate which ballots to misreport. Without this — e.g., if ballots were uniquely identifiable by scanners — it would be straightforward for two scanners to collude to produce incorrect but consistent outputs.
  The ballot indistinguishability assumption is unrealistic for high-resolution scanners that can precisely observe paper fiber patterns or distinctive markings made by voters [10]. However, scanners with in effect one pixel per voting target work well, were once common, and should be compatable with our assumption [35].
  When multiple contests are on the same ballot, malicious scanners could use votes in other contests to distinguish between ballots to coordinate their cheating. This can be entirely prevented by use of a separate paper ballot for each race [4, 40]. Alternatively, we could mask the scanners so that they only observe one one ballot column or ideally just one race, as discussed further in the next bullet and in Section 7. Masking would also limit the ability scanners to use stray marks to trigger cheating.
  That said, the assumption of ballot indistinguishability is perhaps the most problematic assumption we make, and we would recommend further empirical and system design research before any reliance upon it in practice.
- ***Scanner masking assumption*** for 3SCANAUDIT. Our third protocol, 3SCANAUDIT, requires scanners that can be limited to scanning only selected areas of the ballot. In one case, the label but not the vote must be scanned, and in another case, the vote but not the label must be scanned. We further discuss and justify this assumption in Section 7.

[8] The shuffler's intended operation is to mechanically perform a sufficiently entropic, unknown shuffle. Other parts of the protocol ensure we can figure out the permutation that occurred, after this shuffle is performed.

[9] Limiting the trusted hardware to simple, purely mechanical, non-programmable components is desirable because it allows the same hardware to be used without modification for each election, reducing the attack surface.

Algorithm 1 Scan-shuffle-rescan (for C candidates)

procedure $\text{SSR}_C^{S_1,S_2,\Pi}(\mathbf{x})$
  $\mathbf{y} \leftarrow S_1(\mathbf{x})$. *// scanner 1's output (on unshuffled ballots)*
  $(\tilde{\mathbf{x}}, \pi) \leftarrow \Pi(\mathbf{x})$. *// shuffle ballots*
  $\mathbf{z} \leftarrow S_2(\tilde{\mathbf{x}})$. *// scanner 2's output (on shuffled ballots)*
  $d \leftarrow \|\pi(\mathbf{y}) - \mathbf{z}\|_0$. *// # scan discrepancies*
  $\forall c \in [C]$, let $q_c \leftarrow |\{k \in [N] : y_k = c\}|$. *// tallies from* $\mathbf{y}$
  $c_1 \leftarrow \arg\max_{c \in [C]} q_c$. *// winner*
  $c_2 \leftarrow \arg\max_{c \in [C] \setminus \{c_1\}} q_c$. *// runner-up*
  $M \leftarrow q_{c_1} - q_{c_2}$. *// absolute margin*
  return $(\tilde{\mathbf{x}}, \mathbf{y}, \mathbf{z}, \pi, d, M, (q_c)_{c \in [C]})$.
end procedure

Algorithm 2 Basic audit

1: **procedure** $\text{BASICAUDIT}^{S_1,S_2,\Pi}(\mathbf{x}, \alpha)$
2:   $(\tilde{\mathbf{x}}, \mathbf{y}, \mathbf{z}, \pi, d, M, \mathbf{q}) \leftarrow \text{SSR}_2^{S_1,S_2,\Pi}(\mathbf{x})$.
3:   $h \leftarrow \left\lceil \frac{\log(\alpha)}{\log(1-\alpha^{2/M}/2)} \right\rceil$. *// # ballots to hand check*
4:   **if** $d = 0$ **then** *// no discrepancies between two scans*
5:     $\mathbf{v} \leftarrow \text{HANDINSPECT}_h(\tilde{\mathbf{x}})$. *// check h shuffled ballots*
6:     **if** $\forall j \in [h], v_j = z_j$ **then** *// hand check matches scans*
7:       **return** $(\texttt{auto}, \mathbf{y})$.
8:     **end if**
9:   **else** *// one or more discrepancies between scans*
10:     $\mathbf{y} \leftarrow \text{RECOUNT}(\tilde{\mathbf{x}})$. *// full recount*
11:     **return** $(\texttt{hand}, \mathbf{y})$.
12:   **end if**
13: **end procedure**

4 BASICAUDIT

Our simplest model assumes a single two-candidate contest with perfect scanning equipment. Our protocol BASICAUDIT (Algorithm 2) uses two scanners S_1, S_2 and compares the results of the scans. First, S_1 scans the entire set of ballots. Then, the ballots are shuffled randomly before S_2 scans the ballots in shuffled order. These steps make up BASICAUDIT's "scan-shuffle-rescan" or SSR subroutine (Algorithm 1). Our protocols depend on the ***scan discrepancy*** d, the number of ballots that the two scans report differently. BASICAUDIT concludes with manual inspection of a small number h of ballots, and accepts only if the scan discrepancy is zero and hand inspection finds no other discrepancies. Otherwise, BASICAUDIT triggers a full manual recount.

Intuitively, the shuffling step serves to detect adversarial scanner behavior that incorrectly reports only some (but not all) ballots for any given candidate. If the first scanner misreports a subset of the ballots cast for a candidate, it is very unlikely that the second scanner will be able to choose exactly the same subset to misreport if the ballots are presented to the second scanner in random order, unless they both lie on *almost all* or *almost none* of the ballots cast for that candidate. Misreporting almost all ballots for a given candidate is behavior easily detectable by the hand inspection step, and misreporting almost none of the ballots cannot change the outcome unless the margin is very small. Thus, we can conclude in Theorem 1 that BASICAUDIT detects any error in the reported winner with probability $1-\alpha$ for risk limit α. Table 2 gives examples of concrete parameters and the corresponding probabilities of detecting a wrong outcome implied by Theorem 1.

Theorem 1 (BASICAUDIT) *Let S_1, S_2 be scanner functions and let Π be a hiding known-shuffle procedure. Let $\mathbf{x}$ be the ballots cast in a contest. Then $\text{BASICAUDIT}^{S_1,S_2,\Pi}(\mathbf{x}, \alpha)$ outputs the correct winner with probability at least $1-\alpha$.*

In contrast to cryptographic security guarantees, the risk limit α in risk-limiting audits is typically set to be a small constant such as 1% or 10% [25]. However, unlike existing risk-limiting audits, our scheme can also realize cryptographically small risk limits while still requiring hand inspection of only a small number of ballots, as discussed below.

To prove Theorem 1, we establish two important properties of BASICAUDIT. Informally:

- If an adversarial scanner behaves inconsistently on ballot types—i.e., if it assigns some ballots of type a to one reported value and other ballots of type a to another reported value—then SSR will detect this behavior with high probability.
- If a scanner is misreporting a fraction of the true winner's votes, then hand inspecting a small number of ballots will detect this with high probability.

The full proof of the theorem is in Appendix B, where these two properties are formalized as Lemmas 4 and 5.

Table 2 shows the number of ballots that must be hand inspected by BASICAUDIT, for different risk limits α and absolute margins M. As long as the reported margin is larger than 100, it suffices to hand inspect only five ballots to achieve a risk limit of 5%. Even with a reported margin as small as 10, it is sufficient to hand inspect only 10 ballots. These numbers demonstrate the potential power of our approach: with an additional scan, it suffices to hand-inspect an extremely small number of ballots even for very small margins.

Parameter choice The number h of hand-inspected ballots on line 3 of BASICAUDIT is chosen by balancing parameters to guarantee that if at least a $\alpha^{2/M}$ fraction of ballots was misreported by the first scanner, then with probability $1-\alpha$, one of the hand-inspected ballots must have been misreported.

Implementing a known shuffle We propose implementing a known-shuffle procedure using a labeler and an unknown-shuffle (i.e., a mechanical shuffler) as follows. (1) label the ballots $x_1, \ldots, x_N$ with labels $\ell_i = \mathsf{Enc}_K(i)$ where Enc_K denotes encryption with a secret key K; (2) apply an unknown shuffle to the ballots to obtain the ballots in a new order $\tilde{x}_1, \ldots, \tilde{x}_N$; and (3) read and decrypt the labels on the permuted ballots to recover the original index of each permuted ballot, thus

α	Reported absolute margin $M \geq$				
	10	**100**	**1000**	**10^4**	**10^5**
0.09	7	4	4	4	4
0.05	10	5	5	5	5
0.01	21	8	7	7	7
10^{-21}	$*$	230	80	71	70

Table 2: BASICAUDIT number of ballots $h = \lceil \log(\alpha)/\log(1-\alpha^{2/M}/2) \rceil$ to be hand-counted for risk limit α and reported margin M. Starred entries are larger than 1000 and are not recommended for use.

reconstructing the permutation implemented by the shuffle. The use of encryption achieves the *hiding* property required by our protocols (see Section 3); otherwise, simply printing the original indices on the ballots would suffice.

5 2SCANAUDIT

In this section we present a rescan audit for more complicated scenarios that arise in real-world settings with many candidates, ballots audited in separate batches, and imperfect scanners so long as the number of scanner errors does not change the outcome. Since a single election may be administered in many precincts across multiple jurisdictions, it is impractical and possibly illegal to move the ballots to a central location for auditing. Auditing in batches allows the audit to proceed in parallel, allows different ballot designs to be used in different batches, and makes shuffling logistics much simpler by applying the shuffle to smaller sets of ballots.

Unlike in the setting of the previous section, two scans combined and a hand inspection of a small sample of ballots does not suffice to audit the outcome with $C > 2$ candidates or with $B > 1$ batches. This is because the adversary may consistently misreport the votes cast for a candidate who received only a few votes in that batch, which would not be detected in a small sample. To address this problem, we introduce a fixed number t of test ballots for each candidate in each batch. We define the ***test discrepancy*** δ as the number of test ballots misreported by the second scanner. The test ballots serve to ensure that the stack of ballots contains at least a few ballots belonging to each candidate.[10] By checking that the test ballots for each candidate are reported correctly by the second scanner, we can also ensure that the true votes cast for each candidate are reported correctly. This allows us to determine the plurality winner in elections with many candidates and to audit in separate batches. The use of these test ballots obviates the need for a hand inspection after the two scans.

If too many test ballots in any batch are misreported, that batch is manually recounted. Moreover, if reported margin is too small as a function of the risk limit or the discrepancy between the two scans across all batches is large compared to the margin, then all batches are manually recounted.

Algorithm 3 Two-scan audit (single batch)

procedure $\text{BATCH}_{C,B}^{S_1,S_2,\Pi}(\mathbf{x},\alpha,t)$
  $\mathbf{t} \leftarrow 1^{(t)}||\ldots||C^{(t)}$. *// test ballots*
  $\mathbf{x}^+ \leftarrow \mathbf{t}||\mathbf{x}$.
  $(\tilde{\mathbf{x}}^+, \mathbf{y}^+, \mathbf{z}^+, \pi^+, d^+, M^+, \mathbf{q}^+) \leftarrow \text{SSR}_C^{S_1,S_2,\Pi}(\mathbf{x}^+)$.
  $\forall c \in [C], \delta_i \leftarrow |\{i \in [ct] \setminus [(c-1)t] : z_i^+ \neq c\}|$.
  $\delta = \max_c \delta_c$ *// # test discrepancies*
  if $\delta \geq t/10$ **then** *// too many test discrepancies*
    $T \leftarrow \{\pi^+(j)\}_{j\in[Ct]}$. *// test ballots' shuffled indices*
    $\mathbf{x} \leftarrow (\tilde{x}_i^+)_{i\in[Ct+N\setminus T]}$. *// remove test ballots*
    $\mathbf{y} \leftarrow \text{RECOUNT}(\mathbf{x})$. *// recount this batch*
    $\forall c \in [C], q_c \leftarrow |\{k \in [N] : y_k = c\}|$. *// tallies*
    return $(\texttt{hand}, 0, (q_c)_{c\in[C]}, \mathbf{y}, \mathbf{y})$.
  else *// discrepancy small enough: return results w/o recount*
    return $(\texttt{auto}, d, \mathbf{q}, \mathbf{y}, \mathbf{z})$.
  end if
end procedure

Algorithm 4 Two-scan audit (main)

procedure $\text{2SCANAUDIT}_C^{\mathbf{S_1},\mathbf{S_2},\mathbf{\Pi}}((\mathbf{x}_1,\ldots,\mathbf{x}_B),\alpha)$
  Let $t \leftarrow \lceil \frac{25}{8}\log\left(\frac{2BC}{\alpha}\right)\rceil$. *// # test ballots / cand.*
  for $b \in [B]$ **do** *// batch-level audits*
    $(\tau_b, d_b, \mathbf{q}_b, \mathbf{y}_b, \mathbf{z}_b) \leftarrow \text{BATCH}_{C,B}^{S_{1,b},S_{2,b},\Pi_b}(\mathbf{x}_b, t)$.
  end for
  $c_1 \leftarrow \arg\max_{c\in[C]} \left\{\sum_{b\in[B]}(q_b)_c\right\}$. *// winner*
  $c_2 \leftarrow \arg\max_{c\in[C]\setminus\{c_1\}} \left\{\sum_{b\in[B]}(q_b)_c\right\}$. *// runner-up*
  $d \leftarrow \sum_{b\in[B]} d_b$.
  $M \leftarrow q_{c_1} - q_{c_2}$. *// margin*
  if $M \leq \max\{27\cdot\log(2/\alpha), 8d\}$ **then**
    // margin too small: recount
    $\forall b \in [B]$ with $\tau_b = \texttt{auto}$, let $\mathbf{y}_b \leftarrow \text{RECOUNT}(\mathbf{x}_b)$
    $\forall b \in [B]$, let $\tau_b = \texttt{hand}$
  end if
  return $((\tau_1,\ldots,\tau_B), \mathbf{y}_1,\ldots,\mathbf{y}_B)$.
end procedure

Theorem 2 (2SCANAUDIT) *Let S_1, S_2 be scanner functions, let Π be a hiding known-shuffle procedure, and let $\mathbf{x}_1,\ldots,\mathbf{x}_B$ be the ballots cast in batches $1,\ldots,B$ respectively. Then* $\text{2SCANAUDIT}_C^{S_1,S_2,\Pi}((\mathbf{x}_1,\ldots,\mathbf{x}_B),\alpha)$ *will output the correct winner with probability at least $1-\alpha$.*

The proof of Theorem 2 is in Appendix C. While its high-level outline is similar to the proof for BASICAUDIT, the analysis becomes significantly more complex due to the multiple candidates, batching, and imperfect scanners.

Efficiency In addition to satisfying the risk-limit property, it is desirable that the audit procedure only invokes a recount when this is necessary to guarantee the correctness of the outcome.

[10] It may be possible to avoid the use of test ballots if each batch that is not hand-counted can be verified to contain votes cast for every candidate.

			t			
	α	τ_M	$B=1$	$B=20$	$B=200$	$B=2000$
C = 2	**0.09**	83	12	22	29	36
	0.05	99	14	24	31	38
	0.01	143	19	29	36	43
	$\mathbf{10^{-21}}$	1324	156	165	172	180
C = 10	**0.09**	83	17	27	34	41
	0.05	99	19	29	36	43
	0.01	143	24	34	41	48
	$\mathbf{10^{-21}}$	1324	161	170	178	185

Table 3: 2SCANAUDIT recount threshold $\tau_M = \lfloor 27\log(2/\alpha) \rfloor$ on the margin and number $t = \lceil (25/8)\log(2BC/\alpha) \rceil$ of test ballots per candidate per batch with risk limit α, B batches, and C candidates. A full hand recount will be invoked if $M \leq \tau_M$ or $M \leq 8d$.

In 2SCANAUDIT, a single batch is recounted if at least $1/10$ of the test ballots for any candidate are misreported by the second scanner. All batches are recounted if either the overall reported margin is smaller than a function of the risk limit ($27\log(2/\alpha)$) or the number of discrepancies between the two scans is greater than one-eighth of the overall reported margin. Consequently, a recount will only be invoked when there are many misreported ballots or a small margin of victory.

Parameter choices The value of t on line 2 of 2SCANAUDIT is chosen so that with probability $1-\alpha/2$, if the second scanner misreports a majority of the ballots for any candidate in any batch, then at least a small fraction $(1/10)$ of the test ballots will be misreported, violating the test on line 6 of BATCH.[11] As long as the second scanner correctly reports a majority of the ballots for every candidate in every batch, the expected number of discrepancies between the two scans is at least half the number of ballots misreported by the first scanner. The threshold for M on line 10 of 2SCANAUDIT is then chosen to guarantee that with probability $1-\alpha/2$, the number of ballots misreported by the first scanner is smaller than $M/2$.

Table 3 shows the number of test ballots t and threshold margin τ_M for 2SCANAUDIT for various risk limits. We see that the number t of test ballots per batch remains small for a wide range of risk limits α. However, the margin threshold at which our analysis breaks down grows fairly quickly with the number of batches. Improving this dependence for better handling of many batches is a desirable future direction.

6 3SCANAUDIT

The protocols above rely on a known-shuffle procedure (implemented by a labeler and an unknown-shuffle procedure, as noted in Section 4). In this section, we outline a candidate scheme 3SCANAUDIT that we conjecture is secure even in a stronger (and thus preferable) threat model in which *all software components are untrusted*. The trusted components consist solely of simple, non-programmable hardware devices.

[11] Instead of using the fractional threshold $t/10$, we could instead test on line 6 whether any test ballot was misreported. This would also yield a valid audit, but could unnecessarily invoke a recount if a very small number of test ballots are misreported.

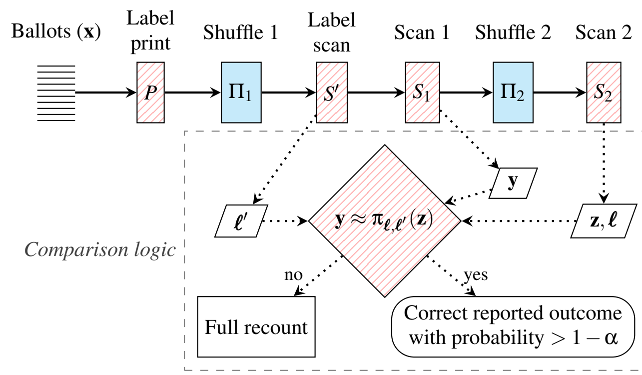

Figure 2: Flow diagram of 3SCANAUDIT. Pink (hatched) components are untrusted. Blue (solid) components are trusted, but purely mechanical (i.e., involve no software). $\pi_{\ell,\ell'}$ denotes the permutation induced by the sequences of labels ℓ, ℓ'.

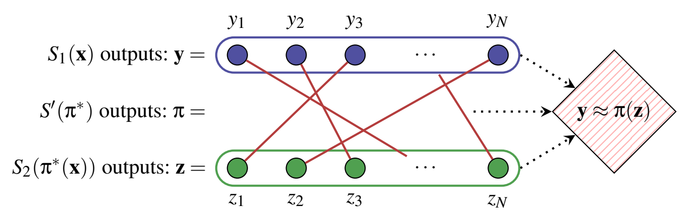

Figure 3: Stylized bipartite graph scheme. S_1, S_2 respectively output lists of scanned vote values $\mathbf{y}, \mathbf{z}$. S' outputs a permutation π (depicted by graph edges) purporting to describe which indices in $\mathbf{y}$ correspond to which indices in $\mathbf{z}$.

Specifically, 3SCANAUDIT would remove the need to trust the labeler, by ensuring that labels are unrelated to the order of ballots in the first scan. Moreover, in contrast to our previous protocols, 3SCANAUDIT's shuffler may be entirely mechanical with no software component, as 3SCANAUDIT requires only an *unknown* shuffle procedure that does not output the permutation implemented. Compared to our earlier protocols, 3SCANAUDIT involves one additional shuffle and one additional scan.

Provable security for 3SCANAUDIT appears substantially more complex, and to require qualitatively new techniques, compared to the analyses of the other schemes in this paper. Providing a complete security proof for 3SCANAUDIT is an open question that we would be keen to see addressed in future work. Below, we give a preliminary security analysis, conjecture the scheme's security, and briefly discuss its overhead and some potential weaknesses. We envision this preliminary analysis to lay useful groundwork for a future security proof; that said, we also view 3SCANAUDIT as a launching point for exploring other variations on the theme.

Candidate scheme 3SCANAUDIT is illustrated in Figure 2: it involves two (unknown) shuffles and three scan procedures. 3SCANAUDIT's innovation is to remove trust in the ballot labeling process and remove the need for a known shuffle, by: (1) shuffling the ballots immediately after labeling (Π_1), before any scans take place; then (2) using scanners to read vote values and labels both before and after the second shuffle (Π_2). 3SCANAUDIT uses test ballots in the same way as 2SCANAUDIT. Intuitively, the first shuffle Π_1 serves to remove any adversarial ordering of labels. The first two scan procedures S', S_1 occur before Π_2: they respectively scan the labels and vote values and are assumed to have physical read access only to the portion of the ballot containing the label or voter markings, respectively.[12] Then, as before, the final scan S_2 occurs after Π_2 and reads both labels and vote values. This results in four scan outputs: ℓ' from S'; $\mathbf{y}$ from S_1; and $(\mathbf{z}, \ell)$ from S_2. Then, these scan results $\mathbf{y}, \mathbf{z}, \ell', \ell$ are checked for consistency by comparison-logic software. Based on ℓ' and ℓ, the comparison logic can compute the permutation π of the ballots between the first and second scan (if the scanners behave honestly), and thus check whether each ballot value in $\mathbf{y}$ is equal to the corresponding value in $\mathbf{z}$. If the scanners behave dishonestly, the inferred permutation π will be incorrect, and this is very likely to cause discrepancies in the comparison logic, as further argued below.

Crucially, this design means that each ballot scan is independent of the ballot permutation π and the ballot labels are independent of the associated vote values. From an entropy perspective, the design is equivalent to the simple "bipartite graph scheme" illustrated in Figure 3 — in which there are no labels, there is one single trusted unknown shuffle, and: S_1, given true ballot values $\mathbf{x}$, outputs scanned values $\mathbf{y}$; S_2, given true shuffled ballot values $\pi^*(\mathbf{x})$, outputs scanned values $\mathbf{z}$; and S', given true permutation π^*, outputs an alleged permutation π. The comparison logic then takes the three outputs $\mathbf{y}, \mathbf{z}, \pi$ and checks whether $\mathbf{y} \approx \pi(\mathbf{z})$, like in 3SCANAUDIT (Figure 2).

Preliminary analysis We now give some intuition behind the conjectured soundness of 3SCANAUDIT. For ease of exposition, the analysis below considers the simpler "bipartite graph scheme" (Figure 3). Let us consider the scans S', S_1, S_2 in turn.

If S' behaves honestly (and the labels are distinct), the inferred permutation π will be correct, so the ballot shuffle amounts to a known-shuffle procedure, and the analysis of 2SCANAUDIT holds. Consequently, any successful attack must have S' output an incorrect label scan.

Now, if S_1 and S_2 behave honestly but S' behaves dishonestly, S' has no information about the outputs of S_1 and S_2. Then, the chance that any incorrect edge that S' produces will pass the comparison logic's consistency check is close to the probability that two randomly sampled ballots have the same value. In close elections, this probability will be close to $1/2$, so the probability that S' can output n incorrect edges and still pass the comparison logic's checks shrinks with $1/2^n$.

The most subtle case to analyze is when S' is dishonest and S_1, S_2 are also dishonest. Then, in the locations where S_1, S_2 output incorrect ballot values that are independent of the true ballot values, S' may be able to compute the outputs of S_1, S_2. However, if any such location is inspected by the hand audit, the hand audit will immediately detect a discrepancy. If the fraction of such locations is ϕ, the probability of evading detection by the hand audit shrinks with $(1-\phi)^h$. Hence, a successful attack must have a relatively small $\phi \ll 1/2$, i.e., not many locations where S_1, S_2 output incorrect values that are independent of the true values.

Finally, if $\phi \ll 1/2$, the ballots on which S_1 output incorrect ballot values that are independent of the true ballot values are very unlikely to be the same physical ballots on which S_2 outputs such incorrect values. That is, there will be many physical ballots for which S_1 outputted an incorrect value but S_2 did not, and vice versa. If S' outputs the correct edge at any one of these locations, then the comparison logic will detect a discrepancy. But if S' outputs an incorrect edge at every such location, then a significant fraction (around $1-\phi^2$) of these incorrect edges must connect to output values on which S_1 or S_2 were honest. Since S' has no information about the output values at these locations, the likelihood that each such edge passes the consistency checks is low (close to $1/2$ in a close election). Therefore, such an attack should be very likely to be detected by the comparison logic whenever S' outputs a significant number of incorrect edges.

Additional considerations for a full analysis It may seem intuitive that the adversary cannot obtain an advantage by outputting an incorrect mapping, since matching up many ballots cast for different candidates makes it likely for discrepancies to be detected. However, this turns out not to be the case: there are nontrivial attacks that involve misreporting in the label scan S', as described in the next paragraph. These attacks are not fatal, but they complicate the rigorous analysis of 3SCANAUDIT and rule out a range of intuitive proof approaches. In particular, bounds on error probabilities for 3SCANAUDIT must be slightly worse than error probabilities for our schemes based on a trusted shuffle, though we expect them still to be exponentially small.

As an illustration, consider a very close election and an adversary wishing to change the outcome by flipping a single vote. If S' is honest, then the probability of both scanners flipping the same vote from the winner to the loser is roughly $1/(n/2) = 2/n$. But if S' observes a sequence of labels beginning with label "k" and misreports by swapping the positions of labels "1" and "k", then S_1 can misreport its first ballot and S_2 can misreport the ballot with label "1". Due to the misreport of S', these ballots are associated in the comparison logic. As long as the ballots labeled "1" and "k" have the same cast vote, no discrepancy will be detected, so this adversary successfully flips a single vote from the winner to the loser in

[12] See Section 7 and "Scanner masking assumption" under Section 3.1 for more discussion on scanner masking.

a close election with probability roughly $1/4$.

Although this attack improves the adversary's chance of flipping a single vote undetected, the probability of flipping t votes undetected decreases exponentially in t. Hence, it fails to provide a meaningful break to the security of the scheme: even in elections with small reported margins, the probability of this attack flipping the outcome undetected is well below any standard risk limit. Yet the existence of such attacks appears to add substantial complexity to the security analysis.

Conjecture 1 *There exist a number of test ballots t and thresholds on the number of misreported test ballots and on the size of the reported margin (independent of the total number of ballots N) such that* 3SCANAUDIT *outputs the correct winner with probability at least* $1-\alpha$.

Challenges of label handling A natural alternative approach for implementing 3SCANAUDIT is to print adhesive labels, shuffle them, and then scan them before affixing them to the edges or margins of ballots — thus more conclusively eliminating the possibility of the label scanner observing the vote values. However, working with loose adhesive labels creates a serious risk of accidental shuffling between the scanning and affixing of the labels. Furthermore, existing label affixing machines peel labels from continuous spools, preventing the labels from being shuffled and requiring us to trust that the order in which they were printed is unknown by the scanners. By printing labels directly on ballots and relying on masking to ensure that the cast votes are hidden during the label scan and labels are hidden during the first vote scan, the implementation discussed above avoids these pitfalls.

Overhead and other considerations While 3SCANAUDIT is designed to remove a significant trust assumption required by 2SCANAUDIT, it does incur some additional costs and drawbacks compared to 2SCANAUDIT (even putting aside the issue of security proofs). The additional scan and shuffle procedures add significant overhead, on the order of hours (see Table 4). Moreover, for any given parameter regime, 3SCANAUDIT will require slightly worse thresholds on the number of misreported test ballots and the size of the reported margin, in order to get the same probability of a correct outcome. Finally, the trust assumptions could be considered incomparable between the two audit schemes, since different kinds of hardware are involved.

7 Practical shuffling

How to shuffle with existing election hardware As discussed in Section 2.1, many scanners include two and sometimes three output hoppers and mechanisms to divert ballots into one or the other. These mechanisms, if controlled by alternative control software, could easily be used for shuffling. A batch of N ballots can be shuffled into an arbitrary permutation with $O(\log N)$ passes through a scanner if each pass randomly diverts ballots into one or the other output hopper before the contents of the hoppers are appended into one stack. This is the reverse of the operation of cutting and shuffling a deck of cards: a "reverse riffle shuffle." Using a pseudorandom number generator to control the diverter, the seed for that generator serves as the key for a known shuffle.

The software controlling the shuffle must not be able to respond to the content of the ballots. If it could, it could create patterns in the output stacks to communicate to the second scan step. To this end, we have explored using peelable adhesive paper (3M Post-it™ notes) as opaque masks to block part of the document glass of a general-purpose high-speed scanner's (Toshiba ES5008A). So long as the mask is placed so the leading edge of the ballot cannot catch the edge of the label, our paper masks worked to block a stripe of any width along the long axis of the ballot without interfering with the paper transport mechanism. While we are confident that scanners could be designed to accomodate better masks, perhaps made of metal shim stock, our experiment demonstrates that our masking assumption is realistic.

An arbitrary shuffle of 1000 ballots using 2 output hoppers requires 10 passes. Most ballot scanners are designed under the assumption that ballots will be counted once and perhaps recounted a very few times. This raises potential concerns about how much wear ballots may suffer if passed through a machine tens of times — concerns augmented by an author's past experience of passing stacks of ballots through a scanner (Optech 4C) up to 24 times, by the end of which the paper-feed rollers of the scanners had left distinct but faint "tire tracks" on the ballots.[13] Other issues raised by multiple passes include the time taken and the likelihood of clerical errors each time the ballots are handled.

To reduce the potential for wear on the ballots and also the time taken to perform a shuffle, we speculate that just one shuffling pass, followed by a "cut," would be sufficient for practical rescan audits. With 3 output hoppers, this gives 1.58 bits of entropy per ballot. For an unknown shuffle, each election observer could be invited to cut the stack.

Labeling Ballot scanners that incorporate a labeler are available. If these are not available, stand-alone labelers are made. For example, one printer designed to add Bates numbers to documents can handle 150 pages per minute at a cost under $5000 [37]. Such machines are designed to print sequential serial numbers; printing an encrypted sequence would need only a small software change.

The labels must be printed on ballots before they are permuted, and must be read by the second scan. Bar codes are not a good fit as they are not human readable—a drawback for transparency in the elections context—so we prefer labeling with human readable numbers in a machine readable font.

[13] More thorough experimentation would, of course, be required to determine how different scanners fare. However, the experience is a useful preliminary indication of the limits of multiple scanning passes, which may impose practical constraints on the design of rescan audits.

A notable advantage of labeling over having the shuffling machine output the permutation directly is that labels are resilient to clerical errors in ballot handling that might unintentionally permute a stack of ballots. A Rhode Island pilot of post-election audits found that unintended rearrangement of ballots was a common problem [19].

Generating randomness To generate pseudorandomness for labeling or a shuffle, it suffices to use a pseudorandom generator based on a key generated with a public dice-rolling ceremony, a common feature of risk-limiting audits in practice (e.g., [19]). Keys must be unknown to the scanners.[14]

Test ballots Test ballots need to be easy for human observers to identify, yet invisible to scanners during an audit. One way to ensure this is to mark the edge of the ballot with a bright color, on an edge that will be masked from view by the scanners, shufflers and other apparatus. Such marking is critical because it makes test ballots readily visible to human observers when they are mingled in a stack of ballots.

8 Evaluation of BASICAUDIT & 2SCANAUDIT

We conduct an evaluation based on cost and timing data from the Rhode Island pilot study of risk-limiting audits [19]. While audit costs are likely to vary significantly, the RI report currently provides the best publicly available documentation on the costs and timings of risk-limiting audits under realistic conditions. The Rhode Island data is likely to provide a better estimate of realistic costs and timings than could experiments run in a research environment.

Our analysis shows that for elections with margins under roughly 1%, rescan audits are competitive with or better than the best known statistical risk-limiting audits in terms of both ***time*** and ***monetary cost*** (provided that the assumptions required for rescan audits are satisfied), as illustrated in Figure 4. An additional advantage of rescan audits is their more ***predictable*** workload, since the workload of rescanning does not depend on the margin, and escalation to a full recount is less likely[16] in a rescan audit than ballot-polling or ballot-comparison audits. Election officials have indicated that a more predictable workload may be preferred even if it is likely to involve more work than a less predictable alternative [19].

Manual ballot inspection We compare the number of ballots that must be handled manually for each audit. For BASICAUDIT and ballot comparison or ballot polling audits, this refers to the number of ballots that are hand-inspected; for 2SCANAUDIT it refers to the number of test ballots. The number of ballots hand examined by a ballot-polling audit (e.g., BRAVO [26]) is an estimated $2\ln(1/\alpha)/m^2$ ballots for a relative margin of $m = M/N$. The number of ballots hand examined by a ballot-comparison audit (e.g., Shangrila [34]) is approximately $1/m$ times fewer, or $2\ln(1/\alpha)/m$ ballots [25].

T_r	**Random seed/key generation**	14m	(one-off)
R_s	**Scan or label ballots**[17]	4,000 ballots/h	
R_{bc}	**Rescan & prepare ballots** (for ballot-comparison audit)	3,240 ballots/h	
R_{bp}	**Rescan & prepare ballots** (for ballot-polling audit)	4,770 ballots/h	
T_{rbc}	**Retrieve a specified ballot** (for ballot-comparison audit)	45s	average
T_{rbp}	**Retrieve a specified ballot** (for ballot-polling audit)	35s 230s	fastest method slowest method
T_{ex}	**Examine a retrieved ballot**	25s	for one contest[18]
T_{opn}	**Open a box of ballots**[19]	15s	

Table 4: Operation timings based on Rhode Island data [19]

Timings We estimate timings for Figure 4 based on the timings of key audit operations as documented in the Rhode Island pilot study [19] (Table 4) and research systematizing the Rhode Island pilot data [5].[20] The Rhode Island study used two ES&S DS850 scanners, whose specifications indicate a processing speed of 300 ballots per minute; however, the pilot study found that "the DS850 tends to jam frequently" and "most of the scanner operator's time was not spent actually scanning the ballots, but handling them before and after the scan," resulting in a 4–5 times slower throughput [19].

The scanning and hand inspection steps in our protocols have direct equivalents in the Rhode Island ballot-comparison pilot, from which timings can be drawn. We estimate *test ballot preparation time* to be 25s, conservatively bounding it by the time to examine a retrieved ballot: if test ballots are machine-produced, then a human will need to examine them; if they are hand-produced, 25s should suffice to fill in a prescribed bubble; and no retrieval is required. We estimate *labeling and shuffling time* by a single pass of all the ballots through a modern ballot scanner such as the DS850. As discussed in Section 7, we envision a "reverse riffle shuffle" followed by cuts, using just a single pass to avoid the prohibitive cost of a fully random shuffle.

Labor costs We estimate labor costs for Figure 4 using a rough estimate of $20 per person-hour and supposing, consistently with the Rhode Island pilot data, that: a scanner operator can operate two scanners at once, teams of two retrieve ballots for inspection, and teams of five examine retrieved ballots.[21] These labor costs do not account for training and equipment.

[14]Key management is as always a challenging task, but may be slightly mitigated by short-lived keys (for the duration of the election).

[16]I.e., a recount is required for a smaller range of margin values.

[17]Estimate from [19, footnote 59].

[18]This timing scales sublinearly for multiple-contest ballots: the average time to examine a ten-contest ballot was 62s. The table omits this figure since our protocols and thus our evaluation focus on the single-contest setting.

[19]As estimated in [5, equation 5.2].

[20]Where applicable, we interpret the RI data favorably for the ballot-polling and ballot-comparison audits (e.g., using 35s, not 230s, for ballot-polling retrieval time)—conservatively evaluating our own protocols in comparison.

[21]We omit the labor cost of the randomness generation step as it is unclear how many paid personnel would be required and the cost is both relatively small and the same for all schemes we consider.

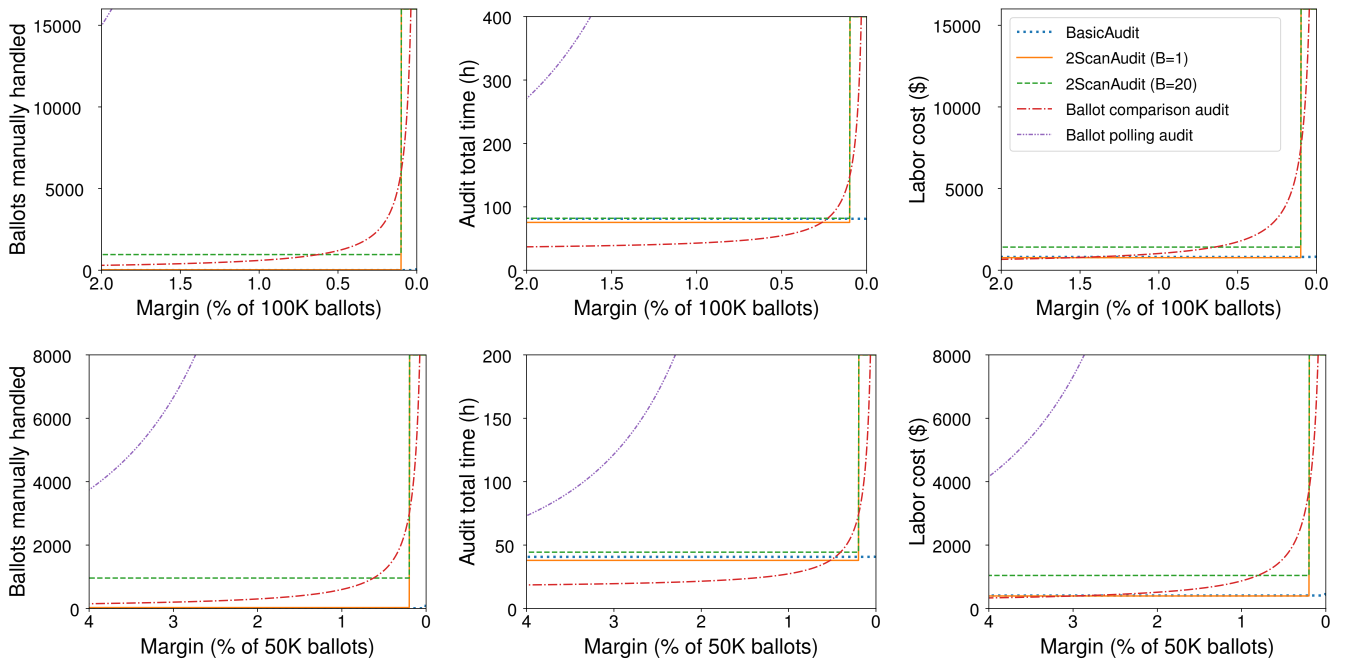

Figure 4: Estimated manual handling (left), total time (center), and labor cost (right) for different election sizes and margins.[15] Workloads are for the case that the audit accepts (i.e., honest scanners). If too many discrepancies are found (or any, in BASICAUDIT), the audit will escalate to a full hand recount. Although 2SCANAUDIT provides better guarantees for $B = 1$ than for $B = 20$, auditing in batches may be desirable due to practical considerations. The formulae used to generate these figures are in Appendix D.

The Rhode Island figures suggest that initial equipment setup may cost roughly \$4,235 per audit location; however, personnel costs are expected to dominate future audit costs, after equipment setup [19].

Comparative worked example. Consider a two-candidate election with $N = 10,000$ ballots cast, relative margin $m = 1\%$ (i.e., absolute margin $M = 100$), and risk limit $\alpha = 5\%$.

- A ***ballot-polling audit*** (e.g., BRAVO [26]) requires hand examining around $2\ln(1/\alpha)/m^2 = 60,000$ ballots. That is, a ballot-polling audit would require a full hand recount in this setting and does not provide any statistical advantage.
- A ***ballot-comparison audit*** (e.g., Shangrila [34]) requires approximately $1/m$ times fewer hand comparisons, leading to a rough estimate of 600 ballots being inspected [25]. We estimate the workload of a ballot-comparison audit to be just short of 15 hours using the formulae in Appendix D. The seed generation takes roughly 14 minutes, the scan and preparation for manual inspection takes about 3.1 hours, and retrieving and manually inspecting the 600 ballots would take about 12.5 hours, for a total of 15.8 hours.
- We estimate the workload of 2SCANAUDIT to be less than 8 hours in total time, using the formulae in Appendix D. The number of test ballots t for each candidate is 14. The seed generation takes roughly 14 minutes, generating the 28 test ballots takes roughly 12 minutes, and the first scan, the label-shuffle step and the second scan each require about 2.5 hours, for a total of less than 8 hours. ***Consequently, in this setting our audit requires less than*** 51% ***of the time of a ballot comparison audit.***

9 Conclusion

We present and analyze new methods for auditing elections, that use untrusted scanners and a very small amount of hand examination of ballots. These methods can handle contests with multiple candidates, ballots that are batched, and error-prone scanners. Our methods are very efficient in the most critical cases, where margins are small. While the schemes proposed here are not ready for near-term deployment, we expect that further theoretical and practical refinements will lead to schemes with an increased domain of practicality.

We recognize that there remain considerable challenges, both theoretical and practical, to our goal of enabling more automation to be used securely in election audits. We hope that the initial steps taken in this paper will guide future research towards making post-election audits both faster and cheaper, while keeping them secure.

[21]The plots depict small margin ranges to illustrate our schemes' performance in the regime where they are competitive. Ballot-polling and ballot-comparison audits perform better for larger margins (not our target regime).

Acknowledgments

We thank Philip Stark for helpful information on timings. SP's research is supported by a Computing Innovation Fellowship, funded by the National Science Foundation under Grant #2127309 to the Computing Research Association, and by the Cornell Tech Digital Life Initiative. SP's earlier work on this project was supported in part by the MIT Digital Currency Initiative. AS is supported by NSF Frontier Award 1804794. Additionally, AS and SP's earlier work on this project was supported by the following grants: NSF MACS (CNS-1413920), DARPA IBM (W911NF-15-C-0236), Simons Investigator award agreement dated June 5th, 2012, and the Center for Science of Information (CSoI), an NSF Science and Technology Center, under grant agreement CCF-093937.

References

[1] Andrew Appel. Optical-scan voting extremely accurate in Minnesota. https://freedom-to-tinker.com/2009/01/21/optical-scan-voting-extremely-accurate-minnesota/, Jan. 21 2019.

[2] Dean Baumert and Mike Dvorak. Ballot processing system. U. S. Patent 8,261,984, Sep. 11 2012.

[3] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 113–131. ACM, 1988.

[4] J. Benaloh, D. Jones, E. Lazarus, M. Lindeman, and P.B. Stark. SOBA: Secrecy-preserving observable ballot-level audit. In *Proceedings 2011 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE '11)*, 2011. http://static.usenix.org/events/evtwote11/tech/final_files/Benaloh.pdf.

[5] Matthew Bernhard. *Election Security is Harder Than You Think*. PhD thesis, U. Michigan, 2020. https://deepblue.lib.umich.edu/handle/2027.42/163272.

[6] J. Bretschneider, S. Flaherty, S. Goodman, M. Halvorson, R. Johnston, M. Lindeman, R.L. Rivest, P. Smith, and P.B. Stark. Risk-limiting post-election audits: Why and how?, Oct. 2012. (ver. 1.1) http://people.csail.mit.edu/rivest/pubs.html#RLAWG12.

[7] Kate Brumback. Georgia again certifies election results showing Biden won. AP News, Dec 2020. https://apnews.com/article/election-2020-joe-biden-donald-trump-georgia-elections-4eeea3b24f10de886bcdeab6c26b680a.

[8] Joseph Calandrino, J. Alex Halderman, and Edward W. Felten. Machine-assisted election auditing. In *Proc. 2007 USENIX/ACCURATE Electronic Voting Technology Workshop*, Aug. 6 2007. https://www.usenix.org/conference/evt-07/machine-assisted-election-auditing.

[9] City and Department of Elections County of San Francisco. Observe the election process. https://sfelections.sfgov.org/observe-election-process [https://perma.cc/3X5L-ETRW].

[10] William Clarkson, Tim Weyrich, Adam Finkelstein, Nadia Heninger, J. Alex Halderman, and Edward W. Felten. Fingerprinting blank paper using commodity scanners. In *2009 30th IEEE Symposium on Security and Privacy*, pages 301–314, 2009.

[11] Colorado Secretary of State Jena Griswold. Risk-Limiting Audit (RLA) FAQs. https://www.sos.state.co.us/pubs/elections/RLA/faqs.html [https://perma.cc/RUH2-W2HN].

[12] Election Assistance Commission. Requirements for the Voluntary Voting system Guidelines 2.0 . https://www.eac.gov/sites/default/files/TestingCertification/Voluntary_Voting_System_Guidelines_Version_2_0.pdf, February 10 2021.

[13] Federal Election Commission. Performance and test standards for punchcard, marksense, and direct recording electronic voting systems. https://josephhall.org/fec_vss_1990_pdf/FEC_1990_Voting_System_Standards.pdf, Jan 1990.

[14] Federal Election Commission. Voting System Standards: Volume I – Performance Standards . https://www.eac.gov/sites/default/files/eac_assets/1/28/Voting_System_Standards_Volume_I.pdf, Apr 2002.

[15] A. Cordero, T. Ji, A. Tsai, K. Mowery, and D. Wagner. Efficient user-guided ballot image verification. In D. Jones, J.-J. Quisquater, and E. Rescorla, editors, *Proceedings 2010 EVT/WOTE Conference*. USENIX/ACCURATE/IAVoSS, August 2010.

[16] European Commission. Eu election missions. http://ec.europa.eu/info/strategy/relations-non-eu-countries/types-relations-and-partnerships/election-observation/mission-recommendations-repository/home [https://perma.cc/KKL4-EU6N].

[17] Florida Division of Elections . Voting System Qualification Test Report – Clear Ballot Group ClearAudit 1.0.6 , 2014. https://files.floridados.gov/media/693729/clear-audit-test-report-112014.pdf.

[18] Georgia Secretary of State. Historic first statewide audit of paper ballots upholds result of presidential race, Nov 2020. https://sos.ga.gov/index.php/elections/historic_first_statewide_audit_of_paper_ballots_upholds_result_of_presidential_race [https://perma.cc/FN4F-V8LE].

[19] Rhode Island RLA Working Group. Pilot implementation study of risk-limiting audit methods in the state of rhode island, aug 2019. https://www.brennancenter.org/sites/default/files/2019-09/Report-RI-Design-FINAL-WEB4.pdf.

[20] Help America Vote Act. https://www.govinfo.gov/content/pkg/PLAW-107publ252/pdf/PLAW-107publ252.pdf, Oct. 29 2002.

[21] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[22] James Hoover, Justin Bales, Steven Bennett, Eric Coomer, Sean Dean, Genice Mathews, and Benjamin Rice. Ballot adjudication in voting systems using ballot images. U. S. Patent 9,202,113, Dec. 1 2015.

[23] Engelbert Hubbers, Bart Jacobs, Berry Schoenmakers, Henk van Tilborg, and Benne de Weger. Description and Analysis of the RIES Internet Voting System. Technical report, Eindhoven Institute for the Protection of Systems and Information, June 24 2008. https://pure.tue.nl/ws/portalfiles/portal/2959981/844708558571577.pdf.

[24] David Levine and Maurice Turner. Restore trust in our democracy through more election transparency. The Hill, 2021. https://thehill.com/opinion/campaign/551760-restore-trust-in-our-democracy-through-more-election-transparency/.

[25] Mark Lindeman and Philip B. Stark. A gentle introduction to risk-limiting audits. *IEEE Security and Privacy*, 10:42–49, 2012.

[26] Mark Lindeman, Philip B. Stark, and Vincent S. Yates. BRAVO: Ballot-polling risk-limiting audits to verify outcomes. In Alex Halderman and Olivier Pereira, editors, *Proceedings 2012 EVT/WOTE Conference*, 2012.

[27] Multiple. Principles and best practices for post-election tabulation audits. https://verifiedvoting.org/publication/principles-and-best-practices-for-post-election-tabulation-audits, December 2018.

[28] National Academies of Sciences, Engineering, and Medicine. *Securing the Vote: Protecting American Democracy*. The National Academies Press, Washington, DC, 2018. https://doi.org/10.17226/25120.

[29] Kathleen O'Neil. Restoring election confidence requires transparency, increased access. American Association for the Advancement of Science, 2018. https://thehill.com/opinion/campaign/551760-restore-trust-in-our-democracy-through-more-election-transparency/.

[30] Election Systems & Software. *ES&S DS850 System Operations Procedures*, 15.0 edition, August 7 2012.

[31] P. B. Stark and D. A. Wagner. Evidence-based elections. *IEEE Security and Privacy*, 10(05):33–41, Sep-Oct 2012.

[32] Philip B. Stark. Conservative statistical post-election audits. *Ann. Appl. Stat.*, 2:550–581, 2008.

[33] Philip B. Stark. Tools for comparison risk-limiting election audits. http://www.stat.berkeley.edu/~stark/Vote/auditTools.htm, 2015.

[34] Philip B. Stark. Sets of half-average nulls generate risk-limiting audits: SHANGRLA, 2020.

[35] James D. Stewart. Device for optically reading marked ballots using infrared and red emitters. U. S. Patent 5,248,872, Sep. 28 1993.

[36] S.W.L. What do election observers do? The Economist. https://www.economist.com/the-economist-explains/2017/06/21/what-do-election-observers-do [https://perma.cc/XHV5-SWHG].

[37] Shear Tech. *DS-6500/6600 Document Sequencer User's Manual*. Chatsworth, CA, 1.3i edition, 2005. https://officeequipmentmachineshop.com/library/DS-6x00-Manual.pdf.

[38] Verified Voting. Audit law database. https://verifiedvoting.org/auditlaws/.

[39] Verified Voting. Post election audits. https://www.verifiedvoting.org/resources/post-election-audits/.

[40] Wikipedia. Elections in South Korea. https://en.wikipedia.org/wiki/Elections_in_South_Korea.

[41] Jan Wolfe. Factbox: Trump's false claims debunked: the 2020 election and jan. 6 riot. Reuters, 2022. https://www.reuters.com/world/us/trumps-false-claims-debunked-2020-election-jan-6-riot-2022-01-06/.

A Tail bounds

Lemma 1 (Hoeffding bound) *Let $X_1, \ldots, X_n$ be independent $[0,1]$-valued random variables, and let $\overline{X} = \frac{1}{n}\sum_i X_i$. Then for any $t \geq 0$, we have that*

$$\Pr[\overline{X} - \mathbf{E}[\overline{X}] \geq t] \leq e^{-2nt^2} .$$

Lemma 2 (Multiplicative Chernoff bound) *Let $X_1, \ldots, X_n$ be independent $\{0,1\}$-valued random variables, let $\overline{X} = \frac{1}{n}\sum_i X_i$ and $\mu = \mathbf{E}[\overline{X}]$. Then for any $\delta \in (0,1)$, we have*

$$\Pr\left[\overline{X} \geq (1-\delta)\mu\right] \leq \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\mu} .$$

Lemma 3 *(Hoeffding [21]) Let $X \sim \mathsf{Hypergeom}(N,K,n)$ be distributed according to the hypergeometric distribution with n samples on a population of size N containing K successes, and let $p = K/N$. Then for any $\zeta > 0$ we have*

$$\Pr[X \leq (p-\zeta)n] \leq e^{-2\zeta^2 n} \text{ and } \Pr[X \geq (p+\zeta)n] \leq e^{-2\zeta^2 n} .$$

B Proofs for BasicAudit

Lemma 4 *Suppose there are k ballots of type (true value) a, and scanner S_1 assigns i of them to one reported value $\hat{x} \in \{1,2\}$ and $k-i$ to the other reported value for $0 < i \leq k/2$. Then* SSR *will output a discrepancy d that is nonzero with probability at least $1-(i/k)^i$.*

Proof: For $j \in \{1,2\}$, let E_j be the event that S_j assigns i type-a ballots to reported value $\hat{x}$ and $k-i$ ballots of type a to reported value $1-\hat{x}$. Suppose (as in the lemma statement) that E_1 occurs. Then if E_2 does not occur, SSR will output a nonzero discrepancy with probability 1.

Now suppose that $E_1 \wedge E_2$ occurs. Consider the set of permutations that agree with π on all ballots not of type a, where π is the permutation sampled by the routine SSR. There are $k!$ such permutations. The input to the second scanner S_2 is identical for each of these permutations. SSR will output a nonzero discrepancy unless S_2 correctly guesses which ballots of type a S_1 assigned to each reported value. Hence, the probability that S_2 agrees with S_1 on each of the ballots of type a is at most $i!(k-i)!/k! = 1/\binom{k}{i} < (i/k)^i$. ■

Lemma 4 immediately yields the following corollary, which implies that SSR will identify a discrepancy with very high probability except in two cases: when almost all ballots cast for the winner are reported correctly (in which case there may be no discrepancy) and when almost all ballots cast for the winner are reported incorrectly.

Corollary 1 *Suppose scanner S_1 reports the wrong vote on at least an ε fraction of the W ballots cast for the true winner, where $\varepsilon \in [0,1]$. Then an incorrect report will be detected — by* SSR *outputting a nonzero discrepancy — with probability at least $1-\hat{\varepsilon}^{W\hat{\varepsilon}}$ where $\hat{\varepsilon} = \min\{\varepsilon, 1-\varepsilon\}$.*

Lemma 5 *Suppose scanner S_1 reports the wrong vote on at least an ε fraction of the ballots cast for the true winner, where $\varepsilon \in [0,1]$. Then an incorrect report will be detected with probability at least $1-(1-\varepsilon/2)^h$ from a hand inspection of h distinct random ballots.*

Proof: The true winner received at least $N/2$ votes, so our assumption implies that S_1 reports the wrong vote on at least $\varepsilon N/2$ ballots. Call these ballots "bad ballots," and call all the others "good ballots." Then the fraction of bad ballots among all ballots is at least $\varepsilon/2$. Consider the sequential selection of h random ballots (with replacement). For each ballot selected, the probability that the selected ballot is good is $1-\varepsilon/2$. It follows that the probability p_{miss} that all of the h ballots selected for hand inspection are good is $(1-\varepsilon/2)^h$. Finally, the probability that at least one of the hand-inspected ballots is bad is $1-p_{\text{miss}}$. ■

Theorem 1 (Correctness of BasicAudit) *Let S_1, S_2 be scanner functions and let Π be a hiding known-shuffle procedure. Let $\mathbf{x}$ be the ballots cast in a contest. Then* BasicAudit$^{S_1,S_2,\Pi}(\mathbf{x},\alpha)$ *outputs the correct winner with probability at least $1-\alpha$.*

Proof: Suppose the reported winner is incorrect and the reported margin is M. Then at least $M/2$ votes for the true winner must have been erroneously reported by scanner S_1 as votes for the true loser. For any $\delta \in [0,1/2]$, consider two cases as follows.

Case I: Less than a δ fraction of the true winner's votes were misallocated by S_1. As noted above, the fraction $\varepsilon < \delta$ of misreported votes is at least $(M/2)/W$, where W is the true number of votes for the true winner. By Corollary 1, the execution of SSR within BasicAudit outputs a zero discrepancy with probability at most

$$\begin{aligned} \varepsilon^{W\varepsilon} &\leq \varepsilon^{M/2} && \text{(since } \varepsilon \geq (M/2)/W\text{)} \\ &\leq \delta^{M/2} . && \text{(since } \varepsilon < \delta\text{)} \end{aligned}$$

Case II: At least a δ fraction of the true winner's votes were misallocated by S_1. In this case, Lemma 5 implies that hand-inspecting h random ballots will detect an error with probability at least $1-(1-\delta/2)^h$.

It follows that BasicAudit outputs the correct winner with probability at least

$$1 - \min_{\eta \in [0,\frac{1}{2}]} \max\left\{\eta^{M/2}, (1-\eta/2)^h\right\} . \quad (1)$$

Taking $\eta = \alpha^{2/M}$, since $h = \lceil \log(\alpha)/\log(1-\alpha^{2/M}/2) \rceil$ in BasicAudit (Algorithm 2, line 3), the theorem follows. ■

C Proofs for 2ScanAudit

Theorem 2 (Correctness of 2ScanAudit) *Let S_1, S_2 be scanner functions, let Π be a hiding known-shuffle procedure, and let $\mathbf{x}_1, \ldots, \mathbf{x}_B$ be the ballots cast in batches $1, \ldots, B$ respectively. Then $2\text{ScanAudit}_C^{S_1,S_2,\Pi}((\mathbf{x}_1, \ldots, \mathbf{x}_B), \alpha)$ will output the correct winner with probability at least $1-\alpha$.*

Proof: Follows from Lemmata 6 and 8. ■

Lemma 6 *Take any batch $b \in [B]$ and any candidate $c \in [C]$. Let $k_{b,c}$ be the true number of votes cast for c in batch b, and let $v_{b,c} \leq t + k_{b,c}$ be the number of the $t + k_{b,c}$ ballots for c in batch b (including test ballots) that that the second scanner in batch b incorrectly reports as being for a candidate other than c. Let E be the event that for every batch b where $\exists c \in [C]$ such that $v_{b,c} \geq (t + k_{b,c})/2$, are manually recounted in the algorithm. Then*

$$\Pr[E] \geq 1 - B \cdot C \cdot \exp(-8t/25) \ .$$

Proof: Let $E_{b,c}$ be the event that $v_{b,c} < (t + k_{b,c})/2$, i.e., the event that less than half of all ballots (including test ballots) of type c in batch b are misreported by the second scanner. Let E_b be the event that either $E_{b,c}$ occurs for all $c \in [C]$, or batch b is manually recounted in the algorithm. Note that $E = E_1 \wedge \cdots \wedge E_B$.

Take any batch $b \in [B]$ and candidate $c \in [C]$ such that $v_{b,c} \geq (t + k_{b,c})/2$. Let R_b be the event that batch b is manually recounted in the algorithm. Let $Q_{b,c}$ be the event that $\delta < t/10$ of the second scanner's misattributed ballots for candidate c in batch b are test ballots. Note that $\neg Q_{b,c} \Rightarrow R_b$.

Conditioned on $\neg E_{b,c}$, we bound the probability of $Q_{b,c}$. Let X be the number of test ballots of type c that the second scanner misreports. Then $X \sim \mathsf{Hypergeom}(t + k_{b,c}, v_{b,c}, t)$ and $\delta \geq X$. Using Lemma 3 with $p = v_{b,c}/(t + k_{b,c})$ and $\zeta = 2/5$:

$$\begin{aligned}
\Pr[Q_{b,c} | \neg E_{b,c}] &= \Pr[\delta < t/10] \leq \Pr[X < t/10] \\
&= \Pr[X < (p - (p - 1/10))t] \\
&\leq \Pr[X < (p - 2/5)t] \qquad (\because p \geq 1/2) \\
&= \Pr[X \leq (p - \zeta)t] \\
&\leq \exp(-2\zeta^2 t) \qquad \text{(by Lemma 3)} \\
&= \exp(-8t/25) = \exp(-0.32t)
\end{aligned}$$

Now returning to analyze E_b, we have

$$\begin{aligned}
E_b &= (E_{b,1} \wedge \cdots \wedge E_{b,C}) \vee R_b \qquad \text{(by definition)} \\
&= (E_{b,1} \vee R_b) \wedge \cdots \wedge (E_{b,C} \vee R_b) \\
&\supset (E_{b,1} \vee \neg Q_{b,1}) \wedge \cdots \wedge (E_{b,C} \vee \neg Q_{b,C}) \quad (\because \neg Q_{b,c} \Rightarrow R_b)
\end{aligned}$$

Using the final expression above to bound $\Pr[E_b]$, we have

$$\begin{aligned}
\Pr[E_b] &> \Pr[(E_{b,1} \vee \neg Q_{b,1}) \wedge \cdots \wedge (E_{b,C} \vee \neg Q_{b,C})] \\
&= 1 - \Pr[(\neg E_{b,1} \wedge Q_{b,1}) \vee \cdots \vee (\neg E_{b,C} \vee Q_{b,C})] \\
&\geq 1 - \sum_{c \in [C]} \Pr[\neg E_{b,c} \wedge Q_{b,c}] \qquad \text{(union bound)} \\
&\geq 1 - \sum_{c \in [C]} \Pr[Q_{b,c} | \neg E_{b,c}] \\
&\geq 1 - C \cdot \exp(-8t/25)
\end{aligned}$$

Finally, we apply another union bound to get

$$\begin{aligned}
\Pr[E] = \Pr[E_1 \wedge \cdots \wedge E_B] &= 1 - \Pr[\neg E_1 \vee \cdots \vee \neg E_B] \\
&\geq 1 - B \cdot C \cdot \exp(-8t/25) \ .
\end{aligned}$$

■

Lemma 7 (Concentration of sums of hypergeometrics) *For $i \in [k]$, let $X_i \sim \mathsf{Hypergeom}(N_i, K_i, n_i)$ be independently hypergeometrically distributed. Let $X = \sum_i X_i$, $n = \sum_i n_i$, and $\mu = \mathbf{E}[X]$. Then for any $\delta \in (0,1)$*

$$\Pr[X < (1-\delta)\mu] \leq \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}} \right)^{\mu}.$$

Proof: Let $N = \sum_i N_i$ and $K = \sum_i K_i$. By construction, $\mu = \mathbf{E}[X] = \sum_i \mathbf{E}[X_i] = \sum_i n_i K_i / N_i$.

Hoeffding [21, Theorem 4] proved that there is a convex order between samples with and without replacement, and that therefore exponential concentration bounds derived via the Cramér-Chernoff method for samples without replacement also apply to samples with replacement. Since $\mathsf{Hypergeom}(N', K', n')$ and $\mathsf{Binom}(n', K'/N')$ differ only in whether the sampling is performed with replacement, in our setting, this means that we can bound X by a sum of binomial distributions with corresponding parameters. That is, $\Pr[X < (1-\delta)\mu] < \Pr[Y < (1-\delta)\mu]$ where $Y = \sum_{i \in [k]} Y_i$ where $Y_i \sim \mathsf{Binom}(n_i, K_i/N_i)$. Since Y is the sum of n independent Bernoulli variables (with different parameters), we can apply a Chernoff bound (Lemma 2) to obtain that

$$\Pr[Y < (1-\delta)\mu] < \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}} \right)^{\mu}.$$

The lemma follows. ■

Lemma 8 *Suppose $t \geq \frac{25}{8} \cdot \log\left(\frac{2BC}{\alpha}\right)$, and let ℓ be the number of ballots misreported by S_1 across all candidates and batches. Then we have that with probability at least $1-\alpha$, either $\ell < 13.04 \log(2/\alpha)$ or $d > \ell/4$.*

Proof: For each batch $b \in [B]$ and candidate $c \in [C]$, let $N_{b,c}$ be the number of ballots in batch b for candidate c (including the t test ballots), and let $K_{b,c}$ and $n_{b,c}$ be the number of these ballots that are misreported by scanners S_1 and S_2, respectively, where we take the convention that $K_{b,c} = n_{b,c} = 0$ for all batches b that were recounted in the algorithm. Then the number of these ballots that are misreported by S_1 and not by S_2 is distributed according to $X_{b,c} \sim \mathsf{Hypergeom}(N_{b,c}, K_{b,c}, N_{b,c} - n_{b,c})$. Let $\mu = \sum_{b,c} \mathbf{E}[X_{b,c}] = \sum_{b,c} \frac{K_{b,c}(N_{b,c}-n_{b,c})}{N_{b,c}}$..

By Lemma 6, we have that with probability at least $1 - B \cdot C \cdot \exp(-8t/25) \geq 1 - \alpha/2$ that each batch for which S_2 misreports at least half of the ballots for any candidate is manually recounted. Condition on the event that this holds. Then we have that $n_{b,c} \leq \frac{N_{b,c}}{2}$ and consequently $\mu \geq \frac{1}{2} \sum_{b,c} K_{b,c} = \ell/2$.

Then by Lemma 7 we can bound the total number of discrepancies by

$$\Pr[d < \ell/4] \leq \Pr[\sum_{b,c} X_{b,c} < \mu/2] \leq (2/e)^{\mu/2}$$

which is at most $\alpha/2$ as long as $\mu \geq 2\log(2/\alpha)/\log(e/2)$, i.e. whenever $\mu \geq 6.52\log(2/\alpha)$. For the case $\mu < 6.52\log(2/\alpha)$, observe that $\ell \leq 2\mu < 13.04\log(2/\alpha)$. Since we have conditioned on an event of probability $1 - \alpha/2$, the conclusion follows by a union bound. ■

D Workload evaluation formulae

N, h, t, C, and B are as defined in BASICAUDIT and 2SCANAUDIT and R_s, R_{bp}, T_{rbp}, R_{bc}, T_{rbc}, T_{ex}, and T_{opn} are as defined in Table 4. BP stands for ballot-polling and BC stands for ballot-comparison.

Here are the formulae we use for Figure 4 timing estimates:

- BP: $T_r + \frac{N}{R_{bp}} + (T_{rbp} + T_{ex}) \cdot 2\ln(1/\alpha)/m^2$.
- BC: $T_r + \frac{N}{R_{bc}} + (T_{rbc} + T_{ex}) \cdot 2\ln(1/\alpha)/m$.
- BASICAUDIT: $T_r + 2 \cdot \frac{N}{R_s} + \frac{N}{R_{bc}} + T_{opn} + h \cdot T_{ex}$.
- 2SCANAUDIT: $T_r + 3 \cdot \frac{N}{R_s} + C \cdot B \cdot t \cdot T_{ex}$.

Let R_L be the cost of labor per person-hour. Here are the formulae we use for Figure 4 cost estimates:

- BP: $R_L(2T_r + \frac{1}{2} \cdot \frac{N}{R_{bp}} + (2T_{rbp} + 5T_{ex}) \cdot 2\ln(1/\alpha)/m^2)$.
- BC: $R_L(2T_r + \frac{1}{2} \cdot \frac{N}{R_{bc}} + (2tT_{rbc} + 5T_{ex}) \cdot 2\ln(1/\alpha)/m)$.
- BASICAUDIT: $R_L(2T_r + \frac{1}{2}\left(2 \cdot \frac{N}{R_s} + \frac{N}{R_{bc}}\right) + T_{opn} + h \cdot 5T_{ex})$.
- 2SCANAUDIT: $R_L(2T_r + \frac{3}{2} \cdot \frac{N}{R_s} + C \cdot B \cdot t \cdot 5T_{ex})$.