UnclearBallot: Automated Ballot Image Manipulation

Matthew Bernhard, Kartikeya Kandula, Jeremy Wink, and J. Alex Halderman

Department of Electrical Engineering and Computer Science, University of Michigan
{matber, kartkand, jreremy, jhalderm}@umich.edu

Abstract. As paper ballots and post-election audits gain increased adoption in the United States, election technology vendors are offering products that allow jurisdictions to review ballot images—digital scans produced by optical-scan voting machines—in their post-election audit procedures. Jurisdictions including the state of Maryland rely on such image audits as an alternative to inspecting the physical paper ballots. We show that image audits can be reliably defeated by an attacker who can run malicious code on the voting machines or election management system. Using computer vision techniques, we develop an algorithm that automatically and seamlessly manipulates ballot images, moving voters' marks so that they appear to be votes for the attacker's preferred candidate. Our implementation is compatible with many widely used ballot styles, and we show that it is effective using a large corpus of ballot images from a real election. We also show that the attack can be delivered in the form of a malicious Windows scanner driver, which we test with a scanner that has been certified for use in vote tabulation by the U.S. Election Assistance Commission. These results demonstrate that post-election audits must inspect physical ballots, not merely ballot images, if they are to strongly defend against computer-based attacks on widely used voting systems.

Keywords: optical scan, paper ballots, image manipulation, drivers, image processing

1 Introduction

Elections that cannot provide sufficient evidence of their results may fail to adequately gain public confidence in their outcomes. Numerous solutions have been posited to this problem [9], but none has been as elegant, efficient, and immediately practical as post-election audits [21, 25, 39]. These audits—in particular, ones that seek to limit the risk of confirming an outcome that resulted from undue manipulation—are one of the most important layers of defense for election security [32].

Risk-limiting audits (RLAs) rely on sampling robust, independent evidence trails created by voter-verified paper ballots. However, other types of post-election

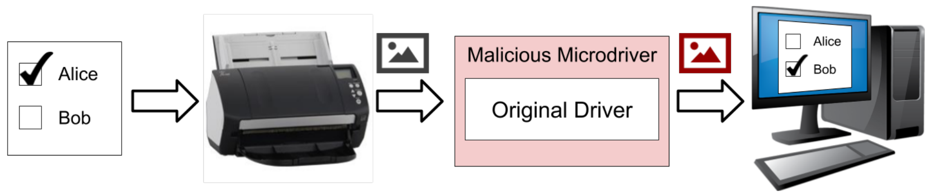

Fig. 1. Attack overview — A voter's paper ballot is scanned by a ballot tabulator, producing a digital image. Malware in the tabulator—in our proof-of-concept, a micro-driver that wraps the scanner device driver—alters the ballot image before it is counted or stored. A digital audit shows only the manipulated image.

audits are gaining popularity in the marketplace. In particular, Clear Ballot, an election technology vendor in the United States, pioneered audit software designed to perform audits of *images* of ballots which have been scanned and tabulated, which we shall refer to as "image audits". Other vendors have adopted support for this kind of audit, and one U.S. state, Maryland, relies on image audits to provide assurances of its election results [33].

While image audits can help detect human error and aid in adjudicating mismarked ballots, we show that they cannot provide the same level of security assurance as audits of physical ballots. Since ballot images are disconnected from the actual source of truth—physical paper ballots—they do not necessarily provide reliable evidence of the outcome of an election under adversarial conditions.

In this paper, we present UnclearBallot, an attack that defeats image audits by automatically manipulating ballot images as they are scanned. Our attack leverages the same computer vision approaches used by ballot scanners to detect voter selections, but adds the ability to move marks from one target area to another. Our method is robust to inconsistent or invalid marks, and can be adapted to many ballot styles.

We validate our attack against a corpus of over 180,000 ballot images from the 2018 election in Clackamas County, Oregon, and find that UnclearBallot can move marks on 34% of the ballots while leaving no visible anomalies. We also test our attack's flexibility using six widely used styles of paper ballots, and its robustness to invalid votes using an established taxonomy of voter marks. As a proof-of-concept, we implement the attack in the form of a malicious Windows scanner driver, which we test using a commercial-off-the-shelf scanner certified for use in elections by the U.S. Election Assistance Commission.

UnclearBallot illustrates that post-election audits in traditional voting systems must involve rigorous examination of *physical ballots*, rather than ballot images, if they are to provide a strong security guarantee. Without an examination of the physical evidence, it will be difficult if not impossible to assure that computer-based tampering has not occurred.

The remainder of this paper is organized as follows: Section 2 provides background on image audits, ballot scanners, and image processing techniques we use to implement our attack. Section 3 describes the attack scenarios against

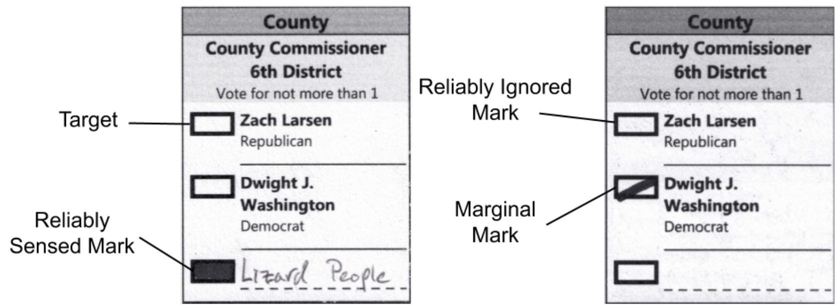

Fig. 2. Terms for parts of a marked ballot, following Jones [23].

optical scanners and image audits. Section 4 explains the methodology of our attack. In Section 5 we present data indicating that our attack can be robust to various ballot styles and voter marks. Section 6 contextualizes our attacks and discusses mitigations. We conclude in Section 7.

2 Background

Our attack takes advantage of two aspects of optical scanner image audits: the scanning and image processing techniques used by scanners, and the reliance on scanned images by image audits. Here we provide a brief discussion of both.

2.1 Ballot Images

Jones [23] put forth an analysis of the way that ballot scanners work, particularly the mark-sense variety that is most common today. All optical scanners currently sold to jurisdictions, as well as the vast majority of scanners used in practice in the U.S., rely on mark-sense technology [44]. Scanners first create a high-resolution image of a ballot as it is fed past a scan head. Software then analyzes the image to identify dark areas where marks have been made by the voter.[1] Once marks have been detected, systems may use template matching to translate marks into votes for specific candidates, typically relying on a barcode or other identifier on the ballot that specifies a ballot style to match to the scanned image.

Detecting and interpreting voter marks can be a difficult process, as voters exhibit a wide range of marking and non-marking behavior, including not filling in targets all the way, resting their pens inside targets, or marking outside the target. The terms Jones developed to refer to the ballot and marks are illustrated in Figure 2. Marks that adequately fill the target and are unambiguously interpreted as votes by the scanner are called *reliably sensed* marks, and targets that are unambiguously not filled and therefore not counted are *reliably ignored* marks.

[1] The details of how marks are identified vary by hardware and scanning algorithm. See [13] for an example.

Fig. 3. Taxonomy of voter marks adapted from Bajcsy [2], including the five leftmost marks that may be considered marginal marks.

Marks of other types are deemed *marginal*, as a scanner may read or ignore them. Moreover, whether a mark should be counted as a vote is frequently governed by local election statute, so some marginal marks may be unambiguously counted or ignored under the law, even if not by the scanner.

Bajcsy et al. [2] further develops a systematization of marginal marks and develops some improvements on mark-detection algorithms to better account for them. An illustration of Bajcsy et al.'s taxonomy is shown in Figure 3. Ji et al. [22] discuss different types of voter marks as applied to write-in votes, as well as developing an automated process for detecting and tabulating write-in selections.

2.2 Image Audits

Risk-limiting post-election audits rely on physical examination of a statistical sample of voter-marked ballots [24, 26, 39, 40]. However, this can create logistical challenges for election officials, which has prompted some to propose relaxations to traditional audit requirements. To reduce workload, canvass audits and recounts in many states rely on retabulation of ballots through optical scanners (see the 2016 Wisconsin recount, for example [31]).

Some election vendors take retabulation audits a step further: rather than physically rescan the ballots, the voting system makes available images of all the ballots for independent evaluation after the election [15, 16, 42].[2] While the exact properties of these kinds of image audits vary by vendor, they typically rely on automatically retabulating all or some images of cast ballots, as well as electronic adjudication for ballots with marginal marks. These "audits" never examine the physical paper trail of ballots, which our attack exploits.

Several jurisdictions have relied on these image audits, including Cambridge, Ontario, which used Dominion's AuditMark [17], and the U.S. state of Maryland, which uses Clear Ballot's ClearAudit [28]. Maryland has also codified image audits into its election code, requiring that an image audit be performed after every election [27].

[2] While the review is made available to the public, the actual images themselves are seldom published in full out of concern for voter anonymity.

3 Attack Scenarios

Elections in which voters make their selections on a physical ballot are frequently held as the gold standard for conducting a secure election [32]. However, the property that contributes most to their security, software independence [34], only exists if records computed by software are checked against records that cannot be altered by software without detection. Image audits enable election officials to view images of ballots and compare them with the election systems' representation of the particular ballot they are viewing (called a cast vote record or CVR). While these two trails of evidence may be independent from each other (for example, Clear Ballot's ClearAudit [15] technology can be used to audit a tabulation performed by a different election system altogether), they are not software independent. A clever attacker can exploit the reliance on software by both evidence trails to defeat detection.

To surreptitiously change the outcome of the election in the presence of an image audit, the attacker must alter both the tabulation result as well as the ballot images themselves. Researchers have documented numerous vulnerabilities that would allow an attacker to infect voting equipment and change tabulation results (see [10, 20, 30] among others), so we focus on the feasibility of manipulating ballot images once an attacker has successfully infected a machine where they are stored or processed.

The most straightforward attack scenario occurs when the ballot images are created by the same equipment that produces the CVR. In this case, the attacker can simply infect the scanner or tabulator with malware that corrupts both the CVR and the images at the same time. The attack could change the image before the tabulator processes it to generate the CVR, or directly alter both sets of records.

In some jurisdictions, the ballot images that are audited are collected in a separate process from tabulation—that is, by scanning the ballots again, as in Maryland's use of ClearAudit from 2016 [28]. In this case, the adversary has to separately attack both processes, and has to coordinate the cheating to avoid mismatches between the initial tally and the altered ballot images.

Depending on the timing of the audit, manipulation of ballot images need not be done on the fly. For example, if the ballot images are created during tabulation but the image audit does not occur until well after the election, an attacker could modify the ballot images while they are in storage.

For ease of explication, the discussion that follows assumes that ballot images are created at the time of tabulation, in a single scan. The attack we develop targets a tabulation machine and manipulates each ballot online as it is scanned.

4 Methodology

To automatically modify ballot images, an attacker can take a few approaches. One approach would be to completely replace the ballot images with ballots filled in by the attacker. However, this risks being detected if many ballots have the same handwriting, and requires sneaking these relatively large data files into the election system without being detected. For these reasons, we investigate an alternative approach: automatically and selectively doctoring the ballot scans to change the vote selections they depict.

For the attack to work successfully, we need to move voter marks to other targets without creating visible artifacts or inconsistencies. We must be able to dynamically detect target areas and marks, alter marks in a way that is consistent with the voter's other marks, and do so in a way that is undetectable to the human eye. However, there is a key insight that works in the adversary's favor: an attacker seeking to alter election results does not have to be able to change *all* ballots undetectably, only sufficiently many to swing the result. This means that the attacker's manipulation strategy is not required to be able to change *every* mark—it merely has to reliably detect *which* marks it can safely alter and change enough of them to decide the election result.

4.1 Reading the ballot

To interpret ballot information, we rely on the same techniques that ballot scanners use to convert paper ballots into digital representations. Attackers have access to the ballot templates, as jurisdictions publish sample ballots well ahead of scheduled elections. Using template matching, an attacker does not have to perform any kind of sophisticated character recognition, they simply have to find target areas and then detect which of the targets are filled.

Our procedure to read a ballot is illustrated in Figure 4. First, we perform template matching to extract each individual race within a ballot. Next, we use OpenCV's [11] implementation of the Hough transform to detect straight lines that separate candidates and break the race into individual panes for each candidate. Notably, the first candidate in each race may have the race title and extra information in it (see Figure 4c), which is cropped out based on white space.

Target areas are typically printed on the ballot as either ovals or rectangles. To detect them, we construct a bounding box around the target by scanning horizontally from the left of the race and then vertically from the bottom up, and compute pixel density values. The bounds are set to the coordinates where the density values first increase and last decrease. Once we have detected all the target areas, we compute the average pixel density of the area within the bounding box to determine whether or not a target area is marked. We then use our template to convert marks into votes for candidates.

4.2 Changing marks

Once we have identified which candidate was marked by the voter, we can move the mark to one of the other target locations we identified. If the vote is for a candidate the attacker would like to receive fewer votes—or if it is not a vote for a candidate they would like to win—the attacker can simply swap the pixels within the bounding boxes of the voter's marked candidate and an unmarked candidate.

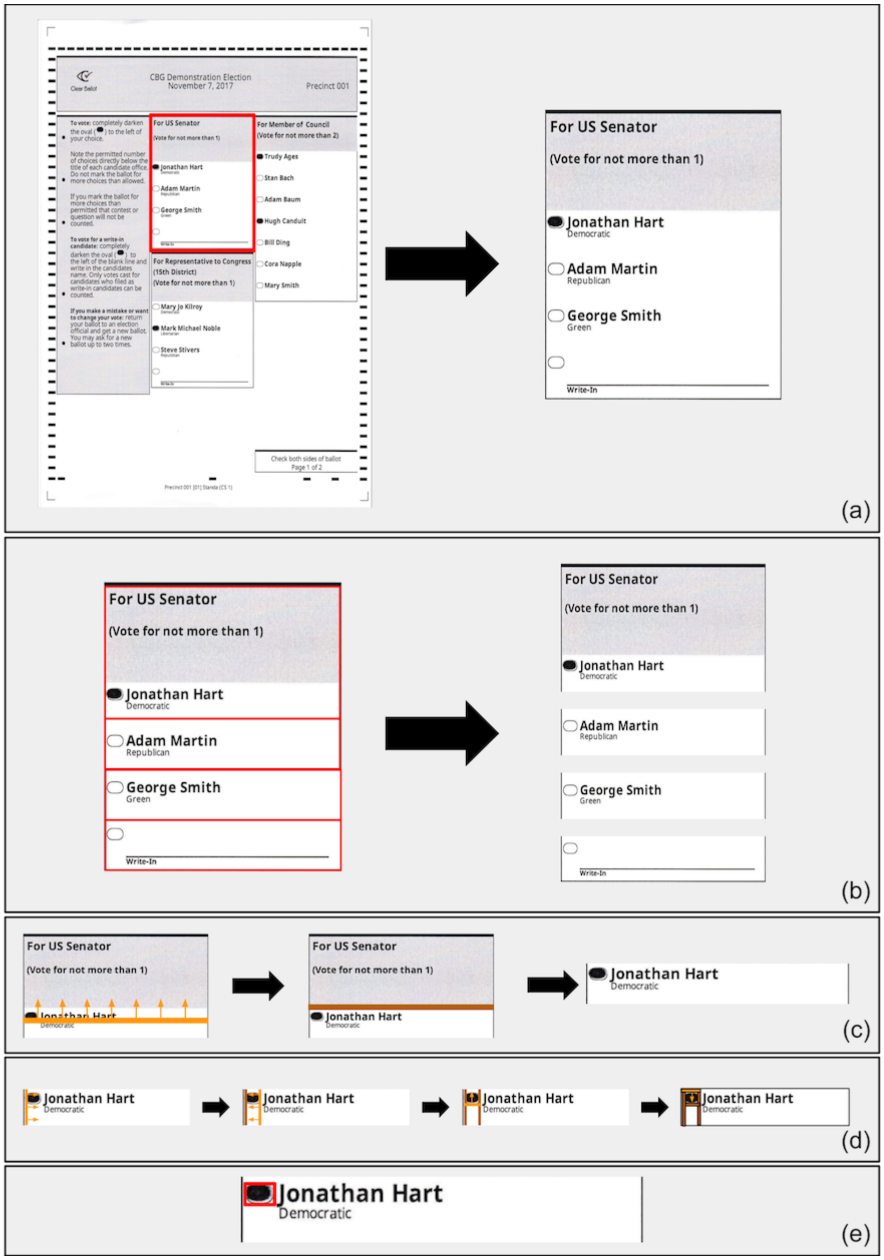

Fig. 4. Ballot manipulation algorithm — First, (a) we apply template matching to extract the race we intend to alter. Then, (b) we use Hough line transforms to separate each candidate. If the first candidate has a race title box, (c) we remove it by computing the pixel intensity differences across a straight line swept vertically from the bottom. For each candidate, (d) we identify the target and mark (if present) by doing four linear sweeps and taking pixel intensity. Finally, (e) we identify and move the mark. At each step we apply tests to detect and skip ballots where the algorithm might leave artifacts.

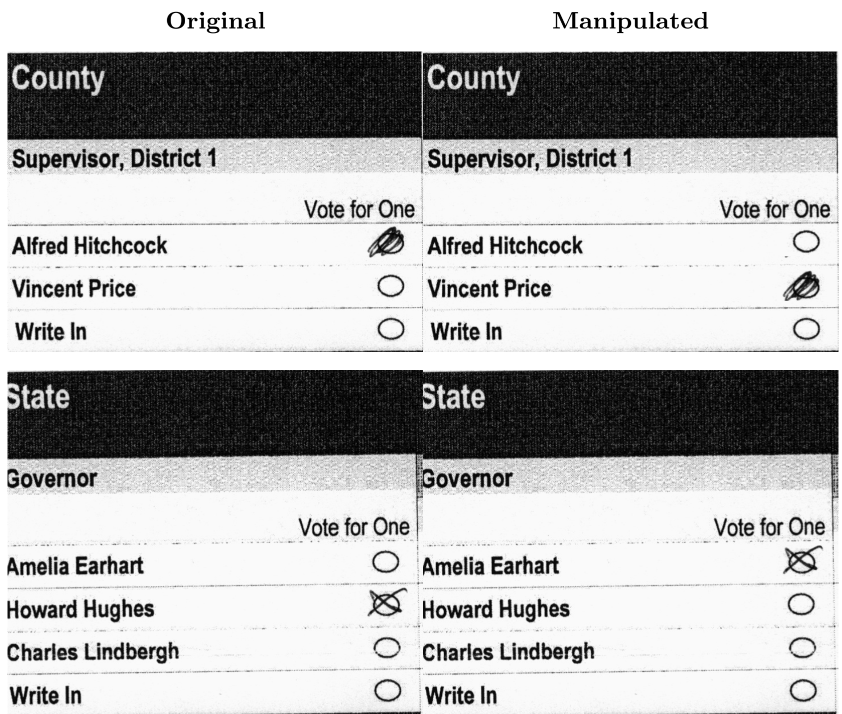

Fig. 5. Automatically moving voter marks — UnclearBallot seamlessly moves marks to the attacker's preferred candidate while preserving the voter's marking style. It is effective for a wide variety of marks and ballot designs. In the examples above, original ballot scans are shown on the left and manipulated images on the right.

By moving marks on each ballot separately, we ensure that the voter's particular style of filling in an oval is preserved and consistent across the ballot. Figure 5 shows some marks swapped by our algorithm, and how the voters original mark is completely preserved in the process.

4.3 UnclearBallot

To illustrate the attack, we created UnclearBallot, a proof-of-concept implementation packaged as a malicious Windows scanner driver, which consists of 398 lines of C++ and Python. We tested it with a Fujistu fi-7180 scanner (shown in Figure 6), which is federally certified for use in U.S. elections as part of Clear Ballot's ClearVote system [43]. These scanners are typically used to handle small volumes of absentee ballots, and must be attached to a Windows workstation that runs the tabulation software.

The UnclearBallot driver wraps the stock scanner driver and alters images from the scanner before they reach the election management application. We chose this approach for simplicity, as the Windows driver stack is relatively easy

Fig. 6. The **Fujitsu fi-7180 scanner** we used to test our attack has been certified by the U.S. Election Assistance Commission for use in voting systems. Our proof-of-concept implementation is a malicious scanner driver that alters ballots on the fly.

to work with, but the attack could also be implemented at other layers of the computing stack. For instance, it could be even harder to detect if implemented as a malicious change to the scanner's embedded firmware. Alternatively, it could could be engineered as a modification to the tabulation software itself.

Once a ballot is scanned, the resulting bitmap is sent to our image processing software, which manipulates the ballot in the way described in Section 4.1. Prior to the election, the attacker specifies the ballot template, which race they would like to affect, and by how much. While ballots are being scanned, the software keeps a running tally of the actual ballot results, and changes ballot images on the fly to achieve the desired election outcome. To avoid detection, attackers can specify just enough manipulated images so that the race outcome is changed.

5 Evaluation

We evaluated the performance and effectiveness of UnclearBallot using two sets of experiments. In the first set of experiments, we marked different ballot styles by hand using types of marks taxonomized by Bajcsy et al. [2]. In the second set of experiments, we processed 181,541 ballots from the 2018 election in Clackamas County, Oregon.

5.1 Testing Across Ballot Styles

In order for our application to succeed at its goal (surreptitiously changing enough scanned ballots to achieve a chosen election outcome), it must be able to detect marks that constitute valid votes as well as distinguish marks which would be noticeable if moved. The marks in the latter case represent a larger set than just marginal marks, as they may indeed be completely valid votes, but considered invalid by our mark-moving algorithm. For example, if we were to swap the targets on a ballot where the user put a check through their target, we may leave a significant percentage of the check around the original target when swapping. The same applies for marked ballots where the filled in area extends into the candidate's name, which could lead our algorithm to swap over parts of the candidate's name when manipulating the image.

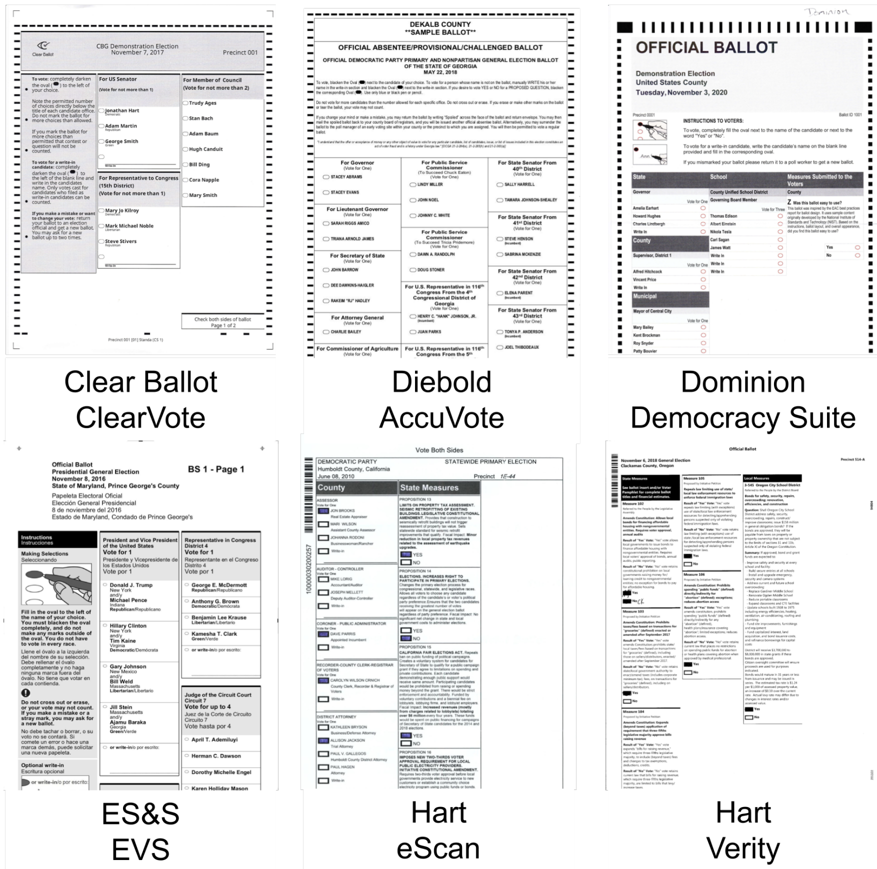

Fig. 7. Ballots Styles — We tested ballot designs from five U.S. voting system vendors: Clear Ballot, Diebold, Dominion, ES&S, and Hart (two styles, eScan and Verity).

To detect anomalies for invalid ballots, we leverage the same intensity checking algorithm that first found the marked areas. The program checks if the width or height is abnormally large, which would indicate an overfilled target, as well as if there are too few or too many areas of high intensity, which would indicate no target or too many targets are filled out. If the program detects an invalid ballot, it will not be modified by the program.

To show our attack is replicable on a variety of different ballot styles, we modified our program to work on six different sample ballot styles, shown in Figure 7. The ballots we tested come from the four largest election vendors in the U.S. (ES&S, Hart InterCivic, Dominion, and Clear Ballot), as well as two older styles of ballots from Hart and Diebold.

Our first experiment was designed to characterize the technique's effectiveness across a range of ballot styles and with both regular and marginal marks. We

Ballot Style	Invalid Marks			Valid Marks			Time/Success
	Skipped	Success	Failure	Skipped	Success	Failure	
Clear Ballot	55	5	0	26	34	0	25 ms
Diebold	60	0	0	6	54	0	11 ms
Dominion	38	22	0	7	53	0	30 ms
ES&S	52	8	0	29	31	0	54 ms
Hart (eScan)	60	0	0	38	22	0	46 ms
Hart (Verity)	60	0	0	27	33	0	21 ms

Table 1. Performance of UnclearBallot — We tested how accurately our software could manipulate voter marks for a variety of ballot styles using equal numbers of invalid and valid marks. The table shows how often the system skipped a mark, successfully altered one, or erroneously created artifacts we deemed to be visible upon manual inspection. We also report the mean processing time for successfully manipulated races, excluding template matching.

prepared 720 marked contests, split evenly among the six ballot styles shown in Figure 7. For each style, we marked 60 contests with what Bajcsy [2] calls "Filled" marks, i.e. reliably detected marks that should be moved by our attack. We marked another 60 ballots in each ballot style with marginal marks, ten each for the five kinds of marginal marks shown in Figure 2 and ten empty marks.

Because the runtime of the template matching step of our algorithm is highly dependent on customization for the particular races on a ballot, we opted to skip it for this experiment. Rather than marking full ballots, we marked cropped races from each ballot style and then ran them through our program. We then manually checked to ensure that the races the program moved were not detectable by inspection. Results for these experiments are shown in Table 1.

Despite rejecting some valid ballots, our program is still able to confidently swap a majority of valid votes. In a real attack, only a small percentage of votes would need to actually be modified, a task easily accomplished by our program. Our program also correctly catches all votes that we have deemed invalid for swapping. This would make it unlikely to be detected in an image audit.

Dominion ballots saw a much higher rate of invalid mark moving, and Diebold and Dominion ballots saw a much higher rate of valid mark moving. This is likely due to the placement of targets: on the Dominion ballots, the mark is right justified, separating it significantly from candidate label information, as can be seen in Figure 7. Similarly, the Diebold ballot provides more space around the target and less candidate information that can be intercepted by marks, which would cause Unclear Ballot to skip moving the mark.

In an online attack scenario (such as if a human is waiting to see the output from the scanner), the attacker needs to be able to modify ballot scans quickly enough not to be noticed. Factors which might affect how quickly our program can process and manipulate ballots include ballot style, layout, and type of mark. During the accuracy experiment just described, we collected timing data for

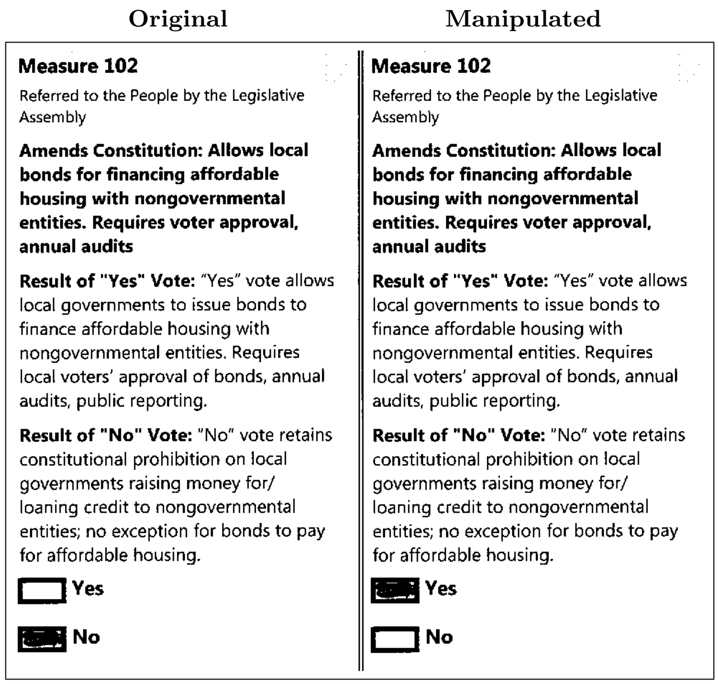

Fig. 8. Attacking Real Ballots — Using 181,541 images of voted ballots from Clackamas County, Oregon, we attempted to change voters' selections for the ballot measure shown above. UnclearBallot determined that it could safely alter 34% of the ballots. For reference, Measure 102 passed by a margin of 5%, well within range of manipulation [14]. We inspected 1,000 of them to verify that the manipulation left no obvious artifacts.

successfully manipulated ballot, and report the results in Table 1. The results show that after the target race has been extracted, the algorithm completes extremely quickly for all tested ballot styles. We present additional timing data at the end of the following section.

5.2 Testing with Real Voted Ballots

To assess the effectiveness of UnclearBallot in a real election, we used a corpus of scans of 181,541 real ballots from the November 6, 2018, General Election in Clackamas County, Oregon, which were made available by Election Integrity Oregon [18]. Like all of Oregon, Clackamas County uses vote-by-mail as its primary voting method, and votes are centrally counted using optical scanners. All images were Hart Verity-style ballots, as shown in Figure 7.

We selected a ballot measure that appeared on all the ballots (Figure 8) and attempted to change each voter's selection. UnclearBallot rejected 20,117 (11%) of the ballots because it could not locate the target contest. We examined a subset of the rejected ballots and found that they contained glitches introduced during scanning (such as vertical lines running the length of the ballot), which interfered with the Hough transform.

To simulate a real attacker, we configured UnclearBallot with conservative parameters, so that it would only modify marks when there was high confidence that the alteration would not be noticeable. As a result, it would only manipulate marks that were nearly perfectly filled in. In most cases, marks that were skipped extended well beyond the target, but the program also skipped undervotes, overvotes, or mislabeled scans. Under these parameters, the program altered the target contest in 62,400 (34%) of the ballot images.

Two authors independently inspected a random sample of 1,000 altered ballots to check whether any contained artifacts that would be noticeable to an attentive observer. Such artifacts might include marks which were unnaturally cut off, visible discontinuities in pixel darkness (i.e. dark lines around moved marks), and so on. If these artifacts were seen during an audit, officials might recheck all of the physical ballots and reverse the effects of the attack. None of the altered ballots we inspected contained noticeable evidence of manipulation.

We also collected timing data while processing Clackamas County ballots. Running on a system with a 4-core Intel E3-1230 CPU running at 3.40 GHz with 64 GB of RAM, UnclearBallot took an average of 279 ms to process each ballot. For reference, Hart's fastest central scanner's maximum scan rate is one ballot per 352 ms [37], well above the time needed to carry out our attack.

These results show that UnclearBallot can successfully and efficiently manipulate ballot images to change real voters' marks. Moreover, the alterations likely would be undetectable to human auditors who examined only the ballot images.

6 Discussion and Mitigations

UnclearBallot demonstrates the need for a software-independent evidence trail against which election results can be checked. It shows that audits based on software which is independent from the rest of the election system is still not software independent. To date, the only robust and secure election technology that is widely used is optical-scan paper ballots with risk-limiting audits based on a robust, well-maintained, *physical* audit trail. However, image audits are not useless, and here we discuss uses for them as well as potential mitigations for our attack.

Uses for image audits. So long as image audits are not the sole mechanism for verifying election results, they do provide substantial benefits to election officials. Using an image audit vastly simplifies some functions of election administration, like ballot adjudication in cases where marks cannot be interpreted by scanners or are otherwise ambiguous. Image audits can be used to efficiently identify and document election discrepancies, as has occurred in Maryland where nearly 2,000 ballots were discovered missing from the audit trail in 2016 [28]. Image audits also identified a flaw in the ES&S DS850 high speed scanner, where it was causing some ballots to stick together and feed two at a time [29].

Another way to utilize image audits is a transitive audit. Methods like SOBA [8] seek to construct an audit trail using all available means of election evidence, rooting the audit in some verification of physical record. By using physical records to verify other records, like CVRs or ballot images, confidence in election outcomes can be transitively passed on to non-physical audit trails. The drawback with this kind of audit is that it usually requires the same level of work as an RLA, plus whatever work is needed to validate the other forms of evidence. However, since ballot image audits already require a low amount of effort, they may augment RLAs and provide better transparency into the auditing process.

Image audits are an augmentation and a convenience for election administration, however, and should not be viewed as a security tool. Only physical examination of paper ballots, as in a risk-limiting audit, can provide a necessary level of mitigation to manipulated election results.

End-to-end (E2E) systems. Voting systems with rigorous integrity properties and tamper resistance such as Scantegrity [12] and Prêt à Voter [35] provide a defense to UnclearBallot. In Scantegrity, when individuals mark their ballots, a confirmation code is revealed that is tied to the selected candidate. This enables a voter to verify that their ballot collected-as-cast and counted-as-collected, as they can look up their ballot on a public bulletin board. Since each mark reveals a unique code, moving the mark would match the code with the wrong candidate, so voters would be unable to verify their ballots. If enough voters complain, this might result in our attack being detected.

Prêt à Voter randomizes the candidate order on each ballot, which creates a slightly higher barrier for our attack, as an additional template matching step would be needed to ascertain candidate order. More importantly, the candidate list is physically separated from the voter's marks upon casting the ballot, so malware which could not keep track of the correct candidate order could not successfully move marks to a predetermined candidate. Since the candidate order is deciphered via a key-sharing scheme, malicious software would have to infect a significant portion of the election system and act in a highly coordinated way to reconstruct candidate ordering. Moreover, as with Scantegrity, votes are published to a public bulletin board, so any voter could discover if their vote had not been correctly recorded.

Other E2E systems which make use of optical scanning and a bulletin board, like STAR-Vote [6], Scratch and Vote [1], and VeriScan [7], are similarly protected from attacks like UnclearBallot.

Other mitigations. Outside of E2E, there may be other heuristic mitigations that can be easily implemented even in deployed voting systems to make our attack somewhat more difficult. As mentioned above, randomizing candidate order on each ballot increases the computation required to perform our attack. Voters drawing outside the bubbles can also defeat our attack, though this might also result in their votes not counting and may be circumvented by replacing the whole race on the ballot image with a substituted one. Collecting ballot images from a different source than the tabulator makes our attack more difficult, as votes now have to be changed in two places. Other standard computer security technologies, like secure file systems, could be used to force the attacker to alter ballot images in a way that also circumvents protections like encryption and permissions.

Detection. Technologies that detect image manipulation may also provide some mitigation. Techniques like those discussed in [3–5, 38], among others, could be adapted to try to automatically detect moved marks on ballots. However, as noted by Farid [19], image manipulation detection is a kind of arms race: given a fixed detection algorithm, adversaries can very likely find a way to defeat it. In our context, an attacker with sufficient access to the voting system to implant a manipulation algorithm would likely also be able to to steal the detector code. The attacker could improve the manipulation algorithm or simply use the detector as part of their mark-moving calculus: if moving a mark will trip the detector, an attacker can simply opt not to move the mark.

While a fixed and automatic procedure for detecting manipulation can provide little assurance, it remains possible that an adaptive approach to detection could be a useful part of a post-election forensics investigation. However, staying one step ahead of sophisticated adversaries would require an ongoing research program to advance the state of the art in detection methods.

A less costly and more dependable way to detect ballot manipulation detection would be to use a software independent audit trail to confirm election outcomes. This can be accomplished with risk-limiting audits, and the software independence enabled by RLAs provides other robust security properties to elections, including defending against other potential attacks on tabulation equipment and servers.

Future work. We have only focused on simple-majority elections here, because those are the kinds of elections used by jurisdictions that do image audits. Audits of more complex election methods, like instant-runoff voting or D'Hondt, have been examined to some extent [36, 41], but future work is needed into audits of these kinds of elections altogether. Because the marks made in these elections are different than the kind we've discussed here, manipulating these ballot images may not be able to employ the same image processing techniques we have used. Additionally it may be difficult for malware to know how many marks it needs to move, since margins in complex elections are difficult to compute. We leave exploration of image manipulation of these elections to future work.

7 Conclusion

In this paper, we demonstrated an attack that defeats ballot image audits of the type performed in some jurisdictions. We presented an implementation using a real scanner, and evaluated our implementation against a set of real ballots and a set of systematically marked ballots from a variety of ballot styles. Our attack shows that image audits cannot be relied upon to verify that elections are free from computer-based interference. Indeed, the only currently known way to verify an election outcome is with direct examination of physical ballots.

Acknowledgements

The authors thank Vaibhav Bafna and Jonathan Yan for assisting in the initial version of this project. They also thank Josh Franklin, Joe Hall, Maurice Turner, Kevin Skoglund, Jared Marcotte, and Tony Adams for their invaluable feedback. We also thank our anonymous reviewers and our shepherd, Roland Wen. This material is based upon work supported by the National Science Foundation under grant CNS-1518888.

References

1. Adida, B., Rivest, R.L.: Scratch and Vote: Self-contained paper-based cryptographic voting. In: ACM Workshop on Privacy in the Electronic Society. pp. 29–40 (2006)
2. Bajcsy, A., Li-Baboud, Y.S., Brady, M.: Systematic measurement of marginal mark types on voting ballots. Tech. rep., National Institute for Standards and Technology (2015)
3. Bayar, B., Stamm, M.C.: A deep learning approach to universal image manipulation detection using a new convolutional layer. In: Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security. pp. 5–10. ACM (2016)
4. Bayram, S., Avcıbaş, İ., Sankur, B., Memon, N.: Image manipulation detection with binary similarity measures. In: 2005 13th European Signal Processing Conference. pp. 1–4. IEEE (2005)
5. Bayram, S., Avcibas, I., Sankur, B., Memon, N.D.: Image manipulation detection. Journal of Electronic Imaging **15**(4), 041102 (2006)
6. Bell, S., Benaloh, J., Byrne, M.D., DeBeauvoir, D., Eakin, B., Fisher, G., Kortum, P., McBurnett, N., Montoya, J., Parker, M., Pereira, O., Stark, P.B., Wallach, D.S., Winn, M.: STAR-vote: A secure, transparent, auditable, and reliable voting system. USENIX Journal of Election Technology and Systems **1**(1) (Aug 2013)
7. Benaloh, J.: Administrative and public verifiability: Can we have both? In: USENIX/ACCURATE Electronic Voting Technology Workshop. EVT '08 (Aug 2008)
8. Benaloh, J., Jones, D., Lazarus, E., Lindeman, M., Stark, P.B.: SOBA: Secrecy-preserving observable ballot-level audit. In: proc. Proc. USENIXAccurate Electronic Voting Technology Workshop (2011)
9. Bernhard, M., Benaloh, J., Halderman, J.A., Rivest, R.L., Ryan, P.Y., Stark, P.B., Teague, V., Vora, P.L., Wallach, D.S.: Public evidence from secret ballots. In: International Joint Conference on Electronic Voting. pp. 84–109. Springer (2017)
10. Bowen, D.: Top-to-Bottom Review of voting machines certified for use in California. Tech. rep., California Secretary of State (2007), https://www.sos.ca.gov/elections/voting-systems/oversight/top-bottom-review/
11. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000)
12. Carback, R., Chaum, D., Clark, J., Conway, J., Essex, A., Herrnson, P.S., Mayberry, T., Popoveniuc, S., Rivest, R.L., Shen, E., Sherman, A.T., Vora, P.L.: Scantegrity II municipal election at Takoma Park: The first E2E binding governmental election with ballot privacy. In: 18th USENIX Security Symposium (Aug 2010)

13. Chung, K.K.t., Dong, V.J., Shi, X.: Electronic voting method for optically scanned ballot (Jul 18 2006), US Patent 7,077,313
14. November 6, 2018 general election. https://dochub.clackamas.us/documents/drupal/f4e7f0fb-250a-4992-918d-26c5f726de3c
15. Clear Ballot: ClearAudit, https://clearballot.com/products/clear-audit
16. Dominion Voting: Auditmark. https://www.dominionvoting.com/pdf/DD%20Digital%20Ballot%20AuditMark.pdf
17. Dominion Voting: Cambridge Case Study. https://www.dominionvoting.com/field/cambridge
18. Election Integrity Oregon, https://www.electionintegrityoregon.org
19. Farid, H.: Digital forensics in a post-truth age. Forensic science international **289**, 268–269 (2018)
20. Feldman, A.J., Halderman, J.A., Felten, E.W.: Security analysis of the Diebold AccuVote-TS voting machine. In: USENIX/ACCURATE Electronic Voting Technology Workshop. EVT '07 (Aug 2007)
21. Hall, J., Miratrix, L., Stark, P., Briones, M., Ginnold, E., Oakley, F., Peaden, M., Pellerin, G., Stanionis, T., Webber, T.: Implementing risk-limiting post-election audits in California. In: 2009 Workshop on Electronic Voting Technology/Workshop on Trustworthy Elections. pp. 19–19. USENIX Association (2009)
22. Ji, T., Kim, E., Srikantan, R., Tsai, A., Cordero, A., Wagner, D.A.: An analysis of write-in marks on optical scan ballots. In: EVT/WOTE (2011)
23. Jones, D.W.: On optical mark-sense scanning. In: Towards Trustworthy Elections, pp. 175–190. Springer (2010)
24. Lindeman, M., Halvorson, M., Smith, P., Garland, L., Addona, V., McCrea, D.: Principles and best practices for post-election audits (Sep 2008), http://electionaudits.org/files/bestpracticesfinal_0.pdf
25. Lindeman, M., Stark, P.: A gentle introduction to risk-limiting audits. IEEE Security and Privacy **10**, 42–49 (2012)
26. Lindeman, M., Stark, P., Yates, V.: BRAVO: Ballot-polling risk-limiting audits to verify outcomes. In: 2011 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections (EVT/WOTE '12). USENIX (2012)
27. Maryland House of Delegates: House Bill 1278: An act concerning election law – postelection tabulation audit. http://mgaleg.maryland.gov/2018RS/bills/hb/hb1278E.pdf
28. Maryland State Board of Elections: 2016 post-election audit report. http://dlslibrary.state.md.us/publications/JCR/2016/2016_22-23.pdf (12 2016)
29. Maryland State Board of Elections: December 15, 2016 meeting minutes. https://elections.maryland.gov/pdf/minutes/2016_12.pdf (Dec 2016)
30. McDaniel, P., Blaze, M., Vigna, G.: EVEREST: Evaluation and validation of election-related equipment, standards and testing. Tech. rep., Ohio Secretary of State (2007), http://siis.cse.psu.edu/everest.html
31. Mebane, W., Bernhard, M.: Voting technologies, recount methods and votes in Wisconsin and Michigan in 2016. 3rd Workshop on Advances in Secure Electronic Voting 2018 (2018)
32. National Academies of Sciences, Engineering, and Medicine: Securing the Vote: Protecting American Democracy. The National Academies Press, Washington, DC (2018), https://www.nap.edu/catalog/25120/securing-the-vote-protecting-american-democracy
33. National Conference of State Legislatures: Post-election audits (January 2019), http://www.ncsl.org/research/elections-and-campaigns/post-election-audits635926066.aspx

34. Rivest, R.: On the notion of 'software independence' in voting systems. Phil. Trans. R. Soc. A **366**(1881), 3759–3767 (October 2008)
35. Ryan, P.Y.A., Bismark, D., Heather, J., Schneider, S., Xia, Z.: Prêt à Voter: A voter-verifiable voting system. IEEE Transactions on Information Forensics and Security **4**(4), 662–673 (2009)
36. Sarwate, A.D., Checkoway, S., Shacham, H.: Risk-limiting audits and the margin of victory in nonplurality elections. Statistics, Politics and Policy **4**(1), 29–64 (2013)
37. ScannerOne: Kodak i5600. http://www.scannerone.com/product/KOD-i5600.html
38. Stamm, M.C., Liu, K.R.: Forensic detection of image manipulation using statistical intrinsic fingerprints. IEEE Transactions on Information Forensics and Security **5**(3), 492–506 (2010)
39. Stark, P.: Conservative statistical post-election audits. Ann. Appl. Stat. **2**(2), 550–581 (2008)
40. Stark, P.: Super-simple simultaneous single-ballot risk-limiting audits. In: 2010 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections (EVT/WOTE '10). USENIX (2010)
41. Stark, P.B., Teague, V., Essex, A.: Verifiable European elections: Risk-limiting audits for D'Hondt and its relatives. USENIX Journal of Election Technology and Systems (JETS) **1**, 18–39 (2014)
42. Unisyn Voting Solutions: OpenElect OCS Auditor. https://unisynvoting.com/openelect-ocs/
43. U.S. Election Assistance Commission: Certificate of conformance: ClearVote 1.5. https://www.eac.gov/file.aspx?A=zgte4IhsHz%2bswC%2bW4LO6PxIVssxXBebhvZiSd5BGbbs%3d (2019)
44. Verified Voting Foundation: The Verifier: Polling place equipment (2019), https://www.verifiedvoting.org/verifier/