Performance Requirements for End-to-End Verifiable Elections

Stefan Popoveniuc, *KT Consulting*
John Kelsey, *NIST*
Andrew Regenscheid, *NIST*
Poorvi Vora, *NIST, GWU*

Abstract

The term "end-to-end verifiability" has been used over the past several years to describe multiple voting system proposals. The term has, however, never been formally defined. As a result, its meaning tends to change from voting system to voting system. We propose a definition for *end-to-end verifiability* of public elections based on *performance requirements*, as opposed to design requirements. We suggest a set of properties that collectively define the term. The properties help detect some of the possible problems that may influence the integrity of the election outcome.

1 Introduction

Over the past decade, several voting system designs have been proposed (e.g. [9],[5],[7],[10],[2],[11],[1],[6]), which exhibit common properties often referred to as *"end-to-end verifiability"*. The designs allow for an open check of the process by which all cast votes are counted correctly. We suggest general performance requirements for end-to-end verifiable elections, focusing on *end-to-end verifiable elections* and not on voting systems: we care if the election outcome accurately reflects the intentions of the voters, regardless of whether the voting equipment is "correct" or not. That is, it is ultimately the election that is checked, not just the equipment.

We consider a simple model of the election process once the voter is authenticated: each voter is presented a ballot and makes a set of choices. He or she then casts this ballot, which contains a representation of the choices (this representation may be an encrypted form of the choices). The representation is recorded, the set of representations tallied, and the tally declared. In an end-to-end verifiable election, it should be possible to check that: the presented ballot is well-formed, the cast ballot is not invalid, the cast ballot is recorded correctly, the tally is correctly computed, the tallied votes are those recorded, and the voting system responds to a voter's instructions as specified by the protocol. The outcome of the numerous possible individual checks—some on individual ballots, several assumed independent of one another—may not always be certain. However, the intention is straightforward: if none of the checks fail, then it should be very unlikely that the reported vote totals are substantially different from the correct totals. We do not attempt to specify the correct probability of detection of a given level of fraud. We do, however, require that some checks provide a high level of certainty, based on what is already available in published end-to-end voting systems.

1.1 Scope

Our definition of end-to-end verification of elections is necessarily very limited. We do not consider fraud originating in the registration database or in permitting unauthorized people to vote – everyone in the graveyard may show up to vote even when end-to-end verification is in use. For all end-to-end voting systems that we are aware of, ballot box stuffing is prevented through the use of procedures. With any of these systems, ballot box stuffing can be detected without reliance on procedures, only if all voters check receipts and this can somehow be reliably known. For example, all voters can digitally sign that they have performed their checks, or all voters sign their receipts. Otherwise, one of the inputs that needs to be checked is the number of active voters. Not knowing the number of active voters leaves the voting system vulnerable to ballot box stuffing.

The scope of end-to-end verification extends from the people who cast votes to the final tally. An election is end-to-end verifiable when any significant difference between the final reported tally and the correct tally that would be computed by counting the ballots that were cast by voters in the system is likely to result in at least one failed check, which produces strong evidence visible to

everyone that the reported result is not correct.

Similarly, we consider requirements only for election integrity. Voter privacy, resistance to coercion and vote buying, reliability, usability, accessibility, and resistance to denial-of-service attacks are all very important in a real-world election, but they are all outside our scope, and will no doubt require many additional requirements. We do realize that the difficulty of designing an end-to-end verifiable system comes from the difficulty of the system providing, in particular, ballot confidentiality. Indeed, it would be rather easy to have an end-to-end voting system in which it is public how everyone voted. It is even more difficult to design an end-to-end verifiable voting system which, in addition to ballot confidentiality, is easy to use, is accessible to voters with disabilities, accommodates all sorts of ballots styles, works in both very small and very large elections, is fast, fail-safe, etc. However, we restrict the scope of this work to rigorously define the notion of *end-to-end verifiability*.

One important note: When we discuss end-to-end verifiability of an *election*, this imposes requirements that go beyond the voting equipment, cryptographic protocols, or programs used. These requirements apply as strongly to the procedures used in these elections, some of which may even be encoded in election law[1].

2 Methodology

To ensure the verifiability of an election, we have identified *six checks*. When a given election passes all six, it can be said that the election is end-to-end verified. Informally, this means that the ballots cast by the voters within the voting system were correctly recorded and counted, and that count was reported in the final tally. If any votes were added[2], deleted, changed, or invalidated after being cast, there is a substantial probability that this fact will be detected in a way that any observer can verify.

Each check is specified by the skeleton below.

1. ***Irregularity* Checked**: Each check is intended to catch a particular type of irregularity.

2. ***When* the check can be made**: It may be that a check can be made only in a window of time; or it may be that a certain event has to happen before the check can be made, but there is no ending time.

3. ***Who* may check**: For a transparent voting system it is unacceptable to allow only entities with privileged access to the voting system or to the election records (like manufacturers, testing labs, election officials) to perform checks. Voters should be able to perform checks to determine their votes are correctly cast. The general public should be able to perform more general checks regarding vote processing. The only requirement for someone to be able to perform checks on the processing of votes should be knowledge of how the system works and access to election data; these should be publicly available.

4. ***What* is checked**: This specifies what it means to perform the check. Perhaps some data is checked for consistency, or perhaps some property of some data is checked.

5. **Detection *Probability***: What is the probability that, if an irregularity has occurred and the check is performed, the check does detect the irregularity? That is, what is $\Pr$[the check detects a problem | there is a problem]? (For general discussions, we denote this probability α, though it should be understood that its value depends on the check being performed). Though it is possible that $\alpha < 1$, and that the check misses the irregularity at times, we would like α to be as close to 1 as possible. The checks should be designed so that, if the election outcome is incorrect, there will be several checks that would independently determine an irregularity with probability α. Hence the probability that no check detected the irregularity would be negligible. Note the following:

   - We assume that the check does not provide a false alarm, *i.e.* $\Pr$[the check detects a problem | there is no problem]$= 0$. Consequently, as long as the check does detect a problem with non-zero probability, *i.e.* $\Pr$[check detects a problem] $\neq 0$, then, if the check indicates there is a problem, there is indeed a problem, *i.e.* $\Pr$[there is no problem | the check detects a problem]$= 0$ (by a simple application of Bayes' rule).
   - While it may be impossible to assess $\Pr$[there is a problem | the check does not detect a problem], this is a probability we care about and which we would like to ensure is small.

6. ***Proof* if system fails check**: If the check detects a problem, is it possible to prove the existence of the problem? How convincing is the proof? Is the proof probabilistic or irrefutable? Is the proof valid for only some entities (e.g. zero-knowledge proofs may only convince the verifier, not anyone else), or can it convince any entity (universal proofs)?

7. **Observations**: Various clarifications for the particular check.

All the requirements are minimal. Some systems may exceed the requirements. For example, suppose there is a requirement that a certain individual should be able to

perform a check. A system in which anyone can perform the check meets and exceeds the requirements. The same is valid for probabilities: systems that provide better than required probabilities of detecting errors meet and exceed the requirements.

Note the following regarding the requirements:

1. No bulletin board is mentioned. While most, if not all, voting systems that do satisfy the end-to-end requirements use a public bulletin board, the definition does not require the use of a bulletin board if the properties may be obtained in another manner.

2. No receipt is mentioned. While some existing end-to-end voting systems give voters a receipt that they can later check is on the public bulletin board, these requirements do not explicitly require that such a receipt exist, but only require the property that is ensured by the use of a receipt.

3. Cryptography is not mentioned. The use or the lack of encryption is not specified by these requirements. While the cast ballot may be an encrypted vote and some cryptographic protocols may need to checked, no reference to any cryptography is made in any of the requirements. The cast ballots may be in the clear or may be encrypted. The tallying mechanism may require decryption or it may not.

4. As mentioned above, the scope of these requirements is only to ensure the integrity of the tally. Ballot secrecy, usability, accessibility and other important requirements of voting systems should also be addressed in separate documents.

5. The requirements do not specify what happens if a problem is found (a system fails a check). It is highly desirable that all systems provide a mechanism for recovery. Assuming a mistake has been detected, it is undesirable that the election come to a halt (or that the only alternative is to restart the election from the very beginning). Many systems only detect errors, but from a practical perspective it would be preferable to have the ability to evaluate the impact of some detected error or fraud in a public way, so that election officials could make an informed decision about whether some failed check called the results of the election into question, or could correct the error without rerunning or invalidating the election. However, such concerns are outside the scope of this paper.

3 Definition

We start with some terminology and notation:

- An *active voter* is one who casts a ballot in an election.
- *Vivian* (the initial "V" is for "voter") is any active voter. Victor is another active voter (any active voter different from Vivian). All voters are equal in the sense that they have the same opportunity to detect an irregularity.
- *Ann* (the initial "A" is for "anyone") is a member of the public, i.e. any person. Ann does not have to be a voter and can be geographically located anywhere. All members of the public are equal in the sense that if Ann has the opportunity to detect an irregularity, Alice has an equivalent opportunity.
- By "*any time after the election*" we mean any time after the polls close and election "housekeeping" is finished (*i.e.* equipment is shut down, reports are made, data is uploaded to servers, *etc.*) This should be no later than a couple of hours after polls close.
- By somebody "*being able to check*" a claim, we mean that the person or entity checking has knowledge of the public election data and of how the voting system works (which should be public). We do not mean that the individual is privileged in any way, and there are no designated or authorized checking entities.
- By *the probability that Vivian detects existent malfeasance*, we mean this probability given that Vivian does perform the check. It is trivial to see that, if Vivian does not opt to perform a check, she is not able to detect malfeasance.
- A *proof* consists of a central claim, and a set of verifiable supporting claims and inferences that demonstrate that the central claim is (probabilistically) true.
- An *irrefutable proof* is a proof in which the probability that the central claim is false is smaller than 2^{-112}. A *probabilistic proof* is one which is not irrefutable. The choice of 2^{-112} comes from NIST's SP 800-57.
- A *publicly acceptable proof* is a proof whose supporting claims and inferences can be independently verified by anyone, as opposed to a proof that is crafted for a designated verifier. If Ann performs a check which fails, and possesses a publicly acceptable proof of malfeasance, she can use the proof to convince Alice of the existence of the malfeasance, or Alice can run the check independently to obtain a proof that convinces her. Note that not all failed checks result in publicly acceptable proofs of malfeasance.

We now provide the definition.

***

An election is said to be end-to-end verifiable if and only if:

1. **Presented ballots are well-formed** (i.e. the representation of the voter's choices on the ballot agrees with the representation that will be read by the rest of the election system) [3]
   (a) ***Irregularity* Checked**: If the ballot to be cast by Vivian is not well-formed, then,
   (b) ***When* the check can be made**: at any time after the election,
   (c) ***Who* may check**: Vivian
   (d) ***What* is checked**: is able to detect if the vote she is about to cast does not represent a vote for the candidate(s) she intended.
   (e) **Detection *Probability***: The probability that Vivian does not detect her incorrectly formed ballot is strictly less than one.
   (f) ***Proof* if system fails check**: Vivian has a publicly acceptable, irrefutable proof of malfeasance if she detects that her ballot is not well-formed.
   (g) **Observations**: If Vivian and Victor are independent samples of voters, then, the probability that Vivian does not detect that her ballot is incorrectly-formed must be independent of the probability that Victor does not detect that his ballot is incorrectly formed. The probability that Vivian does not detect that her ballot is incorrectly formed must be independent of how Vivian voted. Also, Vivian should be able to check that her ballot is "fresh", i.e. that it has not been issued to another voter also.
2. **Cast ballots are well-formed** (*i.e.* cast ballots do not contain over-votes or negative votes)
   (a) ***Irregularity* Checked**: If a cast ballot B (any cast ballot) is not well-formed (contains over-votes or negative votes), and is marked to be included in the tally, then,
   (b) ***When* the check can be made**: at any time after the election,
   (c) ***Who* may check**: Ann
   (d) ***What* is checked**: is able to detect that the cast ballot B is incorrectly formed.
   (e) **Detection *Probability***: The probability that no one detects that the cast ballot B is incorrectly formed is discussed in section 5.
   (f) ***Proof* if system fails check**: Ann has a publicly acceptable, irrefutable proof of malfeasance if she detects that a cast ballot is incorrectly formed.
   (g) **Observations** A "cast ballot" refers to that which has been cast, and which the tally will be computed from. It may not be a plaintext ballot bearing a vote (as in the case of traditional cast paper ballots) and could be an encrypted vote.
3. **Recorded as cast** (i.e., the ballot the voter cast is the one that was received and saved by the voting system)
   (a) ***Irregularity* Checked**: Assuming Vivian's cast ballot has been incorrectly recorded, then,
   (b) ***When* the check can be made**: at any time after the election,
   (c) ***Who* may check**: Vivian
   (d) ***What* is checked**: is able to detect that her[4] cast ballot is incorrectly recorded
   (e) **Detection *Probability***: with a probability strictly greater than zero[5].
   (f) ***Proof* if system fails check**: If Vivian detects that her cast ballot is incorrectly recorded, she has a publicly acceptable proof of malfeasance. The probability that the central claim—the ballot is incorrectly recorded—is true is not negligible. (This proof need not be irrefutable.)
   (g) **Observations**: A ballot is incorrectly recorded if the recorded ballot is different than the ballot that was cast. It should be difficult for Vivian to produce a publicly-acceptable proof of malfeasance, when in fact the vote is recorded correctly (i.e. as cast)[6]. The event corresponding to Vivian checking that her ballot is correctly recorded should be independent of the event corresponding to Victor checking that his ballot is incorrectly recorded, and independent of what vote was cast.
4. **Tallied as recorded** (i.e., the votes on the cast ballots are counted correctly to get the public tally)
   (a) ***Irregularity* Checked**: If n recorded ballots have been incorrectly tallied, then
   (b) ***When* the check can be made**: at any time after the final tally has been made public by the election officials
   (c) ***Who* may check** Ann

(d) ***What* is checked** is able to detect that the declared tally does not represent the tally of all the recorded votes.

(e) **Detection *Probability*** The probability that Ann does not find an error in the declared tally, when there is such an error, is at most $c_n < 1$, a parametric function specified by the design of the system.

(f) ***Proof* if system fails check**: Ann has a publicly acceptable, irrefutable proof of malfeasance if the check fails.

(g) **Observations**: c_n represents the probability of the check passing when n ballots have been mistallied (e.g. $c_n = \frac{1}{2^n}$). The probability c_n may be a function of n, it may be independent of n or it may depend on other parameters such as the total number of cast votes. c_n is known before the election and is specified by the design of the system. For example, c_1 represents the probability of detecting a tally error when a single ballot was miscounted, and c_{20} represents the probability of detecting a tally error assuming 20 ballots were miscounted.

5. **Consistency** (i.e., the set of ballots subject to the *recorded as cast* check is the same as the set of ballots subject to the *tallied as recorded* check.)

   (a) ***Irregularity* Checked**: Assuming that the set of recorded ballots from **Tallied as recorded** is not the same as the set of ballots Vivian is able to check in **Recorded as cast**, then,

   (b) ***When* the check can be made**: at any time after the tally has been made public by the election officials

   (c) ***Who* may check**: Ann

   (d) ***What* is checked**: is able to detect that the two sets are different.

   (e) **Detection *Probability*** The probability that Ann cannot detect that the two sets are different is smaller than ϵ. We suggest $\epsilon = 2^{-30}$, but this may be chosen as desired.

   (f) ***Proof* if system fails check**: Ann has a publicly acceptable irrefutable proof of malfeasance if the system fails the check.

   (g) **Observations**: The "chain of custody" is checked: the cast ballots that voters check are the cast ballots included in the tally.

6. **Each recorded ballot is subject to the "recorded as cast" check** (*i.e.* no ballots are included in the final tally that could not have been checked by at least one voter)

   (a) ***Irregularity* Checked**: If a cast ballot B (any cast ballot) does not have a unique voter who is able to check it during the "recorded as cast" phase, then,

   (b) ***When* the check can be made**: at any time after the election,

   (c) ***Who* may check**: Ann

   (d) ***What* is checked**: is able to detect that the cast ballot B does not have a unique corresponding voter.

   (e) **Detection *Probability***: The probability that no one can detect that n cast ballots do not have corresponding voters should be lower than p_n, where p_n may be a function of n or it may be a low constant (i.e. close to zero).

   (f) ***Proof* if system fails check**: Ann has a publicly acceptable, irrefutable proof of malfeasance if she detects that some ballots do not have unique corresponding voters.

   (g) **Observations** The voting system cannot trick Vivian into checking Victor's ballot while believing she is checking her own ballot, because this would allow the voting system to insert an extra ballot into the tally, without being detected.

In addition to all these checks, there is one additional catch-all requirement, which differs among different end-to-end voting mechanisms too much to be simply specified as a check: Whenever there is some part of the voting protocol which must be followed by the voting system in order to ensure the integrity of the election, there must be some check which can detect (and provide public proof) when the voting system doesn't follow the protocol. For example, if Vivian has a choice of auditing or casting a ballot, and must indicate the choice to the voting system, there needs to be some way for her to prove to others that the voting system ignored her choice. This will often have to be ensured by procedures or physical (non-electronic) mechanisms[7].

If the voting system performs an action which is contrary to what Vivian instructed it to do, then Vivian must have irrefutable proof of malfeasance. For example, if Vivian can choose to either cast or audit a ballot, and Vivian instructs the voting system to audit the ballot, the voting system cannot cast the ballot without Vivian having an irrefutable proof that the voting system did not do what she said.

4 Presented ballots are well-formed

This check ensures that the ballot (whether printed, on screen, or represented in some other way) is interpreted

in the same way by Vivian and by the voting system. Without this check, Vivian might be convinced she had cast a vote for Alice, when her ballot encoded a vote for Bob.

To explain the need for this requirement, we give an example: assume there is an election with a single active voter, Vivian (and thus a single cast ballot). She marked her ballot for Carol (the initial "C" stands for candidate). But because of the special way the ballot was constructed, the final tally contains one vote for Chris. We assume that all the other steps in the process are conducted honestly and all other checks except this one are performed and passed. Vivian's vote was switched from Carol to Chris by the way the ballot is constructed. For example, assume a simple optical scan ballot with two candidates, Carol and Chris. Vivian fills in the oval next to Carol's name. When the piece of paper is inspected, anyone can see that the oval next to Carol is marked and thus this is clearly a vote for Carol. But the optical scanner is configured to detect dark ovals, and not to read the names associated with the dark ovals. A dark oval at some specified geometrical coordinates is counted as a vote for Carol and a dark oval at other specified coordinates for Chris. Here are some possible attacks that the printer can perform:

- print both names on the ballot as Carol (instead of Chris and Carol) and hope that the voter marks the second one, which will be counted for Chris.
- switch the names of the candidates around. Print Carol instead of Chris and Chris instead of Carol.
- translate the entire ballot down, relative to the timing marks of the scanner, such that the geometrical positions the scanner is configured to interpret as for Carol are now for Chris.

The above attacks are for very simple optical scan ballots and are listed in the order of likelihood of being detected by Vivian. It is likely that Vivian will notice that Carol appears twice on the ballot (instead of Carol and Chris), it is less likely that Vivian knows the canonical order of the candidates on the ballot and notices that the ballot she got does not have the candidates in canonical order. It is improbable that Vivian will notice the positions of the races relative to the timing marks.

But not all ballots may be as simple as classical optical scan ballots. In voting systems like Prêt à Voter[7] or PunchScan[10], altering the printing of ballots may not be detectable by a visual inspection. In Prêt à Voter the order of the candidates may not be the committed one, and in PunchScan, the order of the symbols on either the top or the bottom page may be inconsistent with the committed ones. For example, the order of the candidates on the Prêt à Voter ballot that Vivian received would be "Carol, Chris", but the commitment is for the reversed order "Chris, Carol". Vivian can detect this by requesting two ballots, one to spoil and audit and the other one to cast. In PunchScan, the ballot the voter gets may have printed "X" next to Carol and "Y" next to Chris, but the commitment may be to "Y" next to Carol and "X" next to Chris. Vivian can detect this by choosing either the top or the bottom page to keep as her receipt.

It is sufficient if the probability with which Vivian does detect an existing malfeasance is strictly greater than zero, and thus the probability that Vivian does not detect a malformed ballot is strictly less than one. A scenario in which Vivian has no possibility to check if her ballot is well formed is not acceptable.

The probability that both Vivian and Victor fail to detect that their ballots are incorrectly formed is the product of the two probabilities (since they are assumed to be independent of each other). Therefore the probability that several ballots are incorrectly-formed, and none are detected, goes down exponentially with the number of incorrectly-formed ballots. Even a 90% chance of not detecting a malformed ballot translates into a 0.5% chance that 50 ballots were malformed and none were detected (assuming all 50 were checked). For an election with a margin of more than 100 votes, this is a tolerable error rate if all ballots are checked for well-formedness.

To perform this check, Vivian may be asked to compare two strings, two images, or the order in which names appear on a ballot. There may be additional checks, such as checking cryptographic operations (i.e. commitments or encryptions). While Vivian is technically able to do the cryptographic checks, it may be sufficient if Vivian only checks that the data that needs to be cryptographically checked is correct, and lets someone else perform the mathematical checks. For example, in Prêt à Voter, Vivian is asked to compare the order of the candidates on the ballot she spoiled and kept (as a physical piece of paper) to the order of the candidates that is posted on the public bulletin board. Ann can check that the posted order of the candidates is consistent with the decryption onion posted for that ballot.

In addition, Vivian must be able to check that the ballot is receives is "fresh", i.e. that it has not been issues to another voter. This aspect is important to detect the following type of attack: the voting system issues the same ballot to Vivian and Victor and this may in turn cause voters to be associated with the same ballot, allowing the voting system to inject a ballot without being detected.

5 Cast ballots are well-formed

This check ensures that Vivian's vote has the correct impact on the final tally. Thus, Vivian's vote can neither contain multiple votes for her preferred candidate, nor

can contain negative votes to decrease the final count of votes of her least favorite candidate. To best understand this requirement we give two examples.

Assume we have a homomorphic voting scheme like the one in Scratch & Vote[2]. Vivian casts an encrypted vote and all cast votes are first aggregated under the hood of encryption, and then the encrypted aggregate is decrypted, resulting in the tally. There needs to be a check that ensures that the encrypted vote is not an encryption of a value that represents more than one vote for a candidate, or a negative number of votes. For example, if the vector (0,0,1,0) represents a vote for the third candidate and E(0,0,1,0) represents its encryption under the homomorphic scheme, one needs to check that the encryption is not actually for the vector (0,0,100,0) which would count as one hundred votes for the third candidate. Or the vector is not E(-100,-100,301,-100) which would add three hundred and one votes for the third candidate and would deduct 100 votes from all the other candidates, resulting in a net addition of one vote.

Another example is the ThreeBallot[11] voting system. In ThreeBallot, the voter may make three marks for Carol, which would count as an extra vote for Carol, i.e. Vivian casts 2 votes for Carol. Or Vivian may not make any mark for Carol, in which case, Vivian just subtracted one from Carol's total. The ThreeBallot "checker" is designed to avoid these situations.

In most voting systems, under-votes do not actively subtract votes from Carol's total, i.e. an under-vote does not nullify the effect of another valid vote (but in some cases, like ThreeBallot, this is possible). Moreover, over-voted ballots are excluded from the tally. In many cases clear text ballots are available before the tally, so eliminating over-voted ballots is a simple exercise. But in voting schemes that use, for example, homomorphic tallying, clear-text ballots are never available, instead, only the tally is published in clear-text and no clear-text ballots are ever made public.

There are some voting systems in which one can tell if an over-vote or a negative vote is present or not by simply inspecting the cast vote. For example, in Prêt à Voter or PunchScan one can determine, by simply inspecting the coded vote, if the cast ballot should be included in the final tally or not, even before the clear-text votes are available.

Ann should be able to check that no over-votes are included in the final. It is not sufficient if Vivian or the voting system is able to check, because Vivian may be colluding with the voting system. It may only take a single voter colluding with the voting system, to inject a large number of over-votes, thus changing the election outcome.

The potential benefit obtained by inserting a cast ballot that contains 101 votes for Carol is equivalent to stuffing the ballot box with 100 ballots, all for Carol. The risk for the attacker has to be proportional to the benefit. High return should imply a high probability of detection.

We distinguish between two cases:

1. A single detected malformed ballot can be traced back to the voting system (maybe in collusion with Vivian), i.e. the voting system is part of the attack.

2. A single detected malformed ballot can be traced back to Vivian and doesn't call into question the honesty of the voting system.

In the first case, the voting system is an active participant in trying to have over-votes on Vivian's ballot. The probability that Ann does not find an error in this case, when there is such an error, is at most $d_n < 1$, a parametric function specified by the design of the system, where n is the size of the change in the tally caused by the over-voter or negative votes. For example d_n could be lower than $\frac{1}{2^n}$.

In the second case, Vivian is trying to inject more votes than allowed and the voting system is actively trying to stop this from happening. The probability that Vivian can successfully inject over-votes should be negligible, because there may be many voters trying to inject over-votes and it may be enough for one voter to be successful.

The formula $min(2^{-20}, 2^{-max(O,N)})$ where

$$O = log_2(max_{i \in AllCandidates}(NPO_{candidate_i}))$$

and

$$N = log_2(max_{i \in AllCandidates}(|NNV_{candidate_i}|))$$

is an attempt to model this tradeoff. NPO stands for *number of possible overvotes*, and NNV stands for *(*number of negative votes).The more over-votes can be included in a cast ballot, the lower the probability of not detecting this malformed ballot. Similarly, the more votes can be subtracted from a candidate, the lower the probability that this gets unnoticed. This is represented by the term $2^{-max(O,N)}$ Assume that, for a particular form of the cast ballot (e.g. an encrypted ballot for a homomorphic scheme), a voter can hide $2^{100} + 1$ votes for any candidate (that she favors), or -2^{50} votes for any candidate (that she opposes). Then $O = 100$ and $N = 50$ and thus

$$2^{-max(O,N)} = 2^{-max(100,50)} = 2^{-100}$$

There may also be voting systems in which a cast ballot may include a small number of over-votes. ThreeBallot is one example. In ThreeBallot, with the complicity of the ThreeBallot checker, one can cast an extra vote for a candidate or minus one vote for a candidate. This would

result in $O = N = 0$ and thus $2^{-max(O,N)} = 2^0 = 1$. The probability that this ballot is not spotted by the public is 1, which is clearly unacceptable. Therefore we put an upper bound on this probability, 2^{-20}, about one in a million. This completes the explanation of the proposed formula $min(2^{-20}, 2^{-max(O,N)})$.

Note that Ann should be able to perform this check and not only Vivian.

Because the probability of not detecting a malformed ballot is upper bound to at most one in 2^{-20}, and because proof of malfeasance is irrefutable, recovering is possible by simply marking the malformed ballots as not being included in the final tally. This is one of the simplest possible forms of recovery.

6 Recorded as cast

This check ensures that Vivian's vote is recorded by the election system correctly.

This requirement is typically implemented in existing end-to-end systems by giving Vivian a receipt, and posting all receipts on a public bulletin board. Vivian may check that her receipt is correctly posted on the public bulletin board. However, the stated requirement captures a property rather than a mechanism (such as a public bulletin board), and other mechanisms could exist to satisfy it.

This requirement is in preparation for the "Tallied as recorded" requirement, and, in conjunction with the "Consistency" requirement ensures that the chain of custody is secure, from the voters to the talliers. In other words, the ballots that were cast are the ballots that will be tallied. No cast ballot was modified, deleted or substituted (i.e. the original ballot box was not thrown in the river and replaced with a carefully crafted one).

This requirement is typically checked by Vivian having to comparing two strings or two images: one from Vivian's receipt and one provided by the public bulletin board. In essence, the comparison is performed bit by bit, and all bits must be the same. However, it may be that Vivian can only check part of the ballot she cast, not the entire ballot. For example, the cast ballot may be cut into a couple of parts, and Vivian might only be able to check one of the parts, but the voting system may not be able to predict which part before it is checked (e.g as in Three-Ballot). In this case, Vivian probabilistically checks that her ballot is correctly recorded, since she only checks part of her ballot, not all of it.

In most current end-to-end systems, Vivian gets a receipt when she casts her ballot. The receipt can be "something Vivian has", like a paper receipt that is signed (or otherwise authenticated) by the voting system, or the receipt can be "something Vivian knows", such as a secret code that Vivian only learns if she did cast a certain vote. Both types of receipt have advantages and disadvantages. Future end-to-end systems may use some entirely different kind of receipt, so long as they meet these requirements.

If Vivian detects that her ballot is not correctly recorded, she must be able to bring some sort of evidence, so that spurious complaints are discouraged. It is not necessary for Vivian to provide an irrefutable proof of malfeasance, a probabilistic proof is sufficient. For example, if Vivian has a one in thousand chance of forging a proof of malfeasance (as in some settings of Scantegrity II[6]), and a statistically significant number of voters bring such probabilistic proofs, it is unlikely that all were able to successfully forge proofs, and it is very likely that malfeasance occurred. Similarly, if Vivian has a (stamped or digitally signed) receipt that she checks on a public bulletin board, it would be difficult for Vivian to forge the receipt. (It is out of the scope of end-to-end verifiability to describe how Vivian can check the validity of a digital signature. We simply assume that Vivian is able to check the validity of digital signatures, whether procedurally or otherwise).

We note again that this requirement does not protect against ballot box stuffing. It simply ensures that the set of tallied votes includes all the cast votes with very high probability; it does not ensure that there are not votes in the tallied collection that were not legitimately cast. This can be addressed in various ways, for example by making the list of active voters public, or certifying the number of votes cast, *etc.*. We view the voting system as that which counts correctly the votes cast. It is not the system that authenticates voters, or determines who can cast a vote. An end-to-end voting system is not sufficient for a correct election outcome. A means of authenticating voters and determining that votes are cast only by valid voters is also necessary. The issue of ballot box stuffing is outside the scope of this paper.

The fact that Vivian's ballot is incorrectly recorded includes additions, deletions or modifications to the content of the ballot. For example, if Vivian did not vote for anybody, her vote should be recorded as such; it must not be possible for the voting system or election officials to modify this into a different cast vote without her being able to provide (maybe probabilistic) proof that this happened [8] Similarly, it must not be possible for the voting system to modify her valid ballot to be an illegal ballot, e.g. by turning it into an over-vote, without her having some chance of detecting this and being able to provide proof of it.

Conversely, neither Vivian nor Ann should not be able to create a receipt that would falsely accuse the voting system. She should not be able to add, delete or modify any content of her receipt. This is an important requirement to provide resistance of the voting system from a

kind of denial-of-service attack; however, this is outside the scope of a definition of end-to-end election verifiability that considers only integrity.

Vivian should be able to check that HER ballot is recorded as cast and that the ballot is only hers, and not somebody else's also. Many proposed end-to-end voting systems give receipts to each voter and use a public bulletin board to support the "recorded as cast" check. In some systems, it may be possible for the voting system to issue the same receipt to both Vivian and Victor. (Perhaps they voted identically.) In this case, Vivian and Victor each can check that their ballot was recorded as cast. However, since only one public entry was made in the bulletin board for two votes, the voting system is able to introduce an additional entry in the board, encoding whatever vote it likes. If this is possible within a given voting system, there must be a check that detects it with some probability d_n, where n is the number of cast ballots with have a duplicate. More inserted ballots must be no harder to detect than fewer inserted ballots. Note that in many currently proposed end-to-end systems, this consistency check is done by a combination of checking that the ballots are correctly formed, and that there are no duplicate ballot serial numbers included in the set of ballots subject to the final tally.

7 Tallied as recorded

This requirement allows everyone to check that the announced tally has been constructed from all the recorded ballots. Historically speaking, this property is the one that is most studied and most understood. At an abstract level, the voting system may provide a proof of equivalence of two sets, and the proof can be checked by anyone. In practice, techniques such as homomorphic talliers or mix-networks can be used, and it is well understood how such techniques can be made to be universally verifiable.

If Ann detects that a single recorded ballot has been incorrectly included in the tally, then this result must ultimately be available to every observer. That is, there must not be a restricted set of people who can run this check. Instead, if the check fails, it must fail (or be able to fail, given enough trials) for anyone who checks. For some types of systems, such as those that rely on an interactive proof of tally correctness, this requirement may only be satisfied if a beacon of randomness exists, that is accessible by Ann[9].

The upper bound c_n deserves an explanation. c_n may be a function of n (the number of ballots that have been incorrectly recorded) or it may be independent of n. Let's assume there is a winner takes all election, with the margin m. The interesting term is $c_{\frac{m}{2}}$, since at least $\frac{m}{2}$ would have been needed to be changed in order to change the outcome of the election. Let's assume we have two designs of voting systems and c_n is specified for both. We can compare the two designs with regards to this particular check, and can say that one design offers a lower probability of detecting tally errors.

In some existing schemes c_n is an exponential function in n, i.e. a^n where $a < 1$. We can see that $lim_{n\to\infty} a^n = 0, \forall a$ such that $|a| < 1$. Even for $a = 0.9$ (i.e. a very high probability of not detecting that one ballot was not included correctly in the tally) if $c_n = a^n$ then $c_{50} \simeq 0.005$, meaning that the probability of not detecting that 50 ballots were incorrectly included in the tally is approximately 0.5%.

If c_n is not a function of n, then c_n may be a pure constant, or may depend on other tunable parameters. Since c_n does not depend on n, mis-tallying one ballot is as risky as mis-tallying all the ballots. In settings where the expected margin of victory is expected to be very small (*e.g.* under 20 votes), it may be more advantageous if c_n does not depend on n but on another tunable parameter. For example c_n may depend on a parameter d, $c_n = \frac{1}{2^d}$, and d can be set up for a certain election to an arbitrarily high value, such that c_n can be made arbitrarily low. This is the case with zero knowledge proofs for, *e.g.* correct decryption by mixnets.

This requirement does not set an upper threshold for c_n and this decision is left to the election officials. It may be possible to use a different c_n for different elections (even if the same voting system is used), depending on the requirements of the election and the expected margin of victory. It is, however, necessary that $c_n < 1$. That is, if an error occurs, there should be the possibility of detecting it[10]. Further, c_n must at least not *decrease* with n; a larger fraud must not be *less likely* to detect than a smaller one.

This check can be made at any time after the final tally has been made public and certified. Since this check depends on the announced tally, it is not possible to carry it out before the tally is made public. From a technical point of view, there is no time frame to perform this check; more precisely, there is no ending time after which this check cannot be performed anymore. This implies that the results of an election can be contested (mathematically) even after the official tally has been certified and the winner has been installed into office.

8 Consistency

This check ensures that the set of ballots that were subject to the "recorded as cast" check is the same as the set of ballots that were subject to the "tallied as recorded" check.

For example, many proposed end-to-end voting systems use a so-called "public bulletin board". Vivian can

check that her vote is recorded as cast by checking that her receipt is correctly posted on the bulletin board. Anybody can check that all the receipts posted on the public bulletin boards are tallied correctly. If the bulletin board could show a different receipt to Vivian during the recorded as cast check than it shows to Ann during the tallied as recorded check, then the election results could be undetectably altered. Some checks must be done to prevent this; for example, the bulletin board's receipts might be presented online along with a tree signature, which Vivian could use to verify that the data used in the tally included her receipt.

If this check fails, it must fail (or be able to fail, given enough trials) for all observers. That is, if Ann detects a failure, then Alice, Amelia, and Alexis (and any other observer who checks) must also be able to detect it.

9 Each recorded ballot is subject to the "recorded as cast" check

This addresses the following attack: the voting system gives both Vivian and Victor the same receipt. If they both perform the "recorded as cast" check, they may not be able to detect that they are actually checking the same ballot, which would allow the voting system to inject an extra ballot without being detected. This attack is somewhat similar to ballot box stuffing (since an authorized ballot is added), but the stuffing can only be made of one of the valid ballots is first deleted.

Ann must be able to detect if any cast ballot does not have a unique voter who is able to check it during the "recorded as cast" phase. In some voting systems, Ann may inspect the content of the bulletin board and detect that two identical ballots are posted with different data, or with different states. The existing end-to-end voting we are aware of can be divided into two types:

1. the system does not know how the voter wants to vote when handing the voter her ballot.

2. the system knows how the voter votes before giving the voter the receipts.

In the first case (in systems like Prêt à Voter or Punch-Scan), assume Vivian and Victor receive the same ballot, but they vote differently, and thus obtain different receipts. If the system only posts Vivian's receipt, Victor can perform the "Recorded as Cast" check and detect that his receipt is not posted. If the system posts both receipts, this current check allows Ann to detect that there is a duplicate receipt posted.

In the second case, (in systems like Helios), assume Vivian and Victor vote for the same candidate and the system gives them the same receipt. If Vivian chooses to cast her ballot and Victor chooses to audit it, then this current check allows Ann to detect that there is a duplicate receipt posted, and one copy is marked as cast and the other is marked as audited, and thus an inconsistency.

10 Weak versions of requirements

When Vivian detects malfeasance in a check, it is highly desirable that a proof of malfeasance be available. We can imagine the same set of requirements, but without the need for these proofs to exist. For example, in a voting system where Vivian detects that the ballot she is about to cast is incorrectly formed (i.e. detects that the ballot she is about to cast contains a vote for a candidate different than the one she selected) but has no proof, then we say that the weak version of the requirement of a "Ballots are well formed" check is satisfied. Vivian can detect the malfeasance, but has no proof that malfeasance occurred. For all such checks, Vivian must still be able to detect the malfeasance. The only weak aspect is Vivian's ability to produce (probabilistic or irrefutable) proof about the existence of the malfeasance.

In general, an election in which a failure in the above checks always leads to a public proof of malfeasance is better than one in which some failures leave no public proof. However, some kinds of malfeasance may be extremely difficult to detect in a way that provides proof in all cases.

11 Examples

We discuss several voting systems that have been proposed by the community[11] as end-to-end voting systems, and briefly analyze their conformance to the performance requirements outlined in this document. Note that traditional paper ballot systems, optical scan systems and DREs clearly do not satisfy the performance requirements. The obvious requirement which is not met is *Consistency*, because, even if votes are counted in public, Vivian cannot check that the votes being counted are exactly the ones that were cast.

We find that some proposed systems do conform, some need small modifications, some conform to weaker versions of some requirements, and some do not conform to some of the requirements. While we briefly described the main idea of each system, a detailed description of these systems is out of the scope of this paper. The set of voting systems studied is certainly not exhaustive.

Two of the requirements are satisfied in the same way by all end-to-end voting systems that we are aware of, so we describe them here. These voting systems give voters receipts, which are posted on a public bulletin board. Each voter can check that his or her receipt is correctly posted on the public bulletin board, and each can bring

the receipt as proof if they notice that the posting is not correct. The cast ballots are actually the receipts obtained by voters, which can be viewed as encryptions of their votes. This approach satisfies the requirement for the"Recorded as Cast" and, arguably, the "Consistency" checks.

First, we discuss the check for "Recorded as Cast": Vivian should be able to check that her cast ballot is correctly posted. Since Vivian gets a receipt for the ballot she cast, if her cast ballot has been incorrectly recorded, then, after her receipt has been published on the public bulletin board, Vivian is able to detect that her ballot is incorrectly recorded by checking if the receipt she possesses is correctly posted. If Vivian detects that her ballot is incorrectly recorded, then Vivian has her receipt which must serve as a publicly-acceptable proof of malfeasance. It may serve as irrefutable proof of malfeasance it is a "something you have" receipt that is somehow authenticated by the accused voting system, or it may serve as a probabilistic proof if it is a "something you know" receipt. The only system that currently uses the "something you know" receipt is Scantegrity II. We assume that all "something you have" receipts are authenticated by the voting system (digitally signed or rubber stamped), and that Vivian cannot home-brew a valid receipt.

Second, the "Consistency" requirement translates into Ann being able to detect if the bulletin board gives different views to different people. In other words, the public bulletin board does not give some information to Vivian, who checks if her receipt is correctly posted, and some other information to Ann, who checks that all the receipts have been correctly tallied in the "Tallied as Recorded" requirement. No current implementation of such a bulletin board exists, but we assume it is sufficient to have a simple web page that serves the entire information from the bulletin board each time it is accessed. This approach can be made somewhat more secure by having a distributed bulletin board.

Another common aspect of all the systems we consider is that ballots have a unique serial number. A potential attack on the "Ballots are well formed" requirement is to have two ballots with the same serial number. This may be detected by voters, since they may fill in their ballots differently, and thus obtain two different receipts with the same serial number. If both voters perform the "Recorded as cast" check, one voter will detect that her receipt is incorrectly posted.

Finally, all the examples below allow Vivian to detect if the voting system gives the same receipt to her and to Victor. Two general mechanisms can be used. First, it may be possible to publish the names of the voters next to each of the receipts on the public bulletin board. Each receipt must only have one name, thus associating two voters with the same receipt would be detectable by Ann. Second, each ballot may be uniquely identified by, say, a serial number. If the voting system gives a ballot with the same serial number to both Vivian and Victor, than, it may happen that Vivian votes differently than Victor does, and thus Vivian would get a receipt which looks different from Victor's receipt. If both Victor and Vivian check the public bulletin board, one of them can see that his or her receipt is incorrectly posted. Thus the problem reduces to how well can the voting system predict that two voters will vote in the same way. In addition, Vivian may audit her ballot to see if it is well formed, and this check may reveal that she got a duplicate ballot.

11.1 Prêt à Voter

Prêt à Voter [7] is a paper-based system that uses a two-part ballot. The left part contains the names of the candidates in a permuted order and the voter can make a mark in the right part, next to her favorite candidate. The order of the candidates on the left part is different for every ballot. To produce her receipt, the voter detaches and shreds the left side, while scanning and keeping the right part as a receipt. The random ordering of candidates is obtained by permuting the list of candidates in a canonical order. This permutation is a composition of multiple sub-permutations, each sub-permutation being derived from a seed. The seeds are buried into a series of digital envelopes which are processed using a mixnet to obtain the tally.

Presented ballots are well-formed An incorrect Prêt à Voter ballot has the candidates printed in an order that is different from the one derived from the seeds. To check that her ballot is well formed, Vivian can choose a number of ballots, one to vote and the rest to audit. The audited ballots have their digital envelopes opened and the seeds extracted, so that Vivian can check that the permutation of candidates printed on the ballot is indeed derived from the seeds. Typically, a voter may choose two ballots, one to vote and one to audit, in which case the probability that Vivian does not detect that her ballot is incorrectly formed is 50%. The "Presented ballots are well formed" requirement is satisfied.

Cast ballots are well-formed Since the Prêt à Voter cast ballot cannot contain any negative votes, and since anyone can detect over-votes by simply inspecting the receipts published on the public bulletin board, the requirement for a check for "Cast ballots are well formed" is satisfied.

Tallied as recorded Prêt à Voter uses a publicly-verifiable onion mixnet, so that Ann can check that all the cast ballots have been "Tallied as recorded". Typically, $c_n = \frac{1}{2^n}$ if randomized partial checking is used.

Each recorded ballot is subject to the "recorded as cast" check If two voters are given the same ballot, it may happen that they choose different candidates thus Ann can detect that the same receipt is posted twice on the bulletin board, with different selections.

In conclusion, the elections which use Prêt à Voter are end-to-end verifiable.

11.2 PunchScan

A PunchScan [10] ballot consists of two stacked sheets of paper. The top page of the ballot has holes in it, and the information on the bottom page can be read through the holes. The top page contains the candidates' names in a fixed order. Each candidate has a symbol assigned to it, and the assignment of symbols to candidates varies from ballot to ballot. On the bottom page, there is a list of the same symbols, in an order that also differs from ballot to ballot, and is independent of the order on the top page. The top and the bottom ballot pages are aligned in such a way that the symbols from the bottom page are visible through the holes.

The voter uses a dauber to mark the hole that contains the symbols corresponding to her favorite candidate. The voter chooses one of the two pages to keep as receipt, and the other page is shredded. The receipt is scanned and signed before being given to the voter. To count the votes, PunchScan used a two-mix concept that is based on commitments.

Presented ballots are well-formed To check that the ballots are properly formed, a two-stage process is used. First, anybody can check that, with high probability, for every ballot, the combination of the two permutations from the two pages of a PunchScan ballot is inverted by the two mixes used in counting the votes. This is done by challenging the PunchScan system to open the commitments to the two permutations printed on the ballot, as well as to the two permutations used by the two mixes, and checking that their composition is the identity. Second, Vivian is able to check that the ballot she gets is consistent with the commitments to the two permutations, by selecting one of the two pages at random to keep as her receipt. Since PunchScan does not know which page Vivian is keeping as a receipt, the system has a 50% chance of cheating and not being detected by Vivian. If Vivian detects that the permutation on her receipt is not consistent with the commitment, then Vivian's receipt serves as irrefutable proof of malfeasance. The requirement for the "Presented ballots are well-formed" check is satisfied.

Cast ballots are well-formed The PunchScan cast ballot cannot contain any negative votes. An over-vote for a race is visible to anyone who inspects the receipts published on the public bulletin board. Therefore the "Cast ballots are well formed" requirement is satisfied.

Tallied as recorded PunchScan uses a publicly-verifiable two mix mechanism to check that all the cast ballots have been "Tallied as recorded". Typical values for c_n are either $c_n = \frac{1}{2^n}$ if randomized partial checking[8] per ballot is used, or $c_n = \frac{1}{2^d}$, if multiple pairs of mixes are partially checked (where d is a fixed parameter). Therefore the requirement for a "Tallied as recorded" check is satisfied.

Each recorded ballot is subject to the "recorded as cast" check If two voters are given the same ballot, it may happen that they choose different pages to keep and thus Ann can detect that the same ballot have both pages posted on the bulletin board.

In conclusion, the elections which use PunchScan are end-to-end verifiable.

11.3 Scratch&Vote

For simplicity we refer to the Scratch&Vote [2] variant that uses a Prêt à Voter ballot style. Scratch&Vote adds to the Prêt à Voter ballot a 2D bar code that contains all the possible encrypted votes a voter may choose. By making a mark on the right side of the ballot, the voter is actually choosing the corresponding encrypted vote that is embedded in the bar code. There is a scratch-off surface below the 2D bar code that has printed underneath the information (e.g. randomness or keys) that allows the reconstruction of the encryption. As in Prêt à Voter, Vivian marks the right side and separates it, while destroying the left side. Vivian scans and keeps the right side as her receipt (but without the scratch-off surface).

Presented ballots are well-formed To check that the 2D bar code correctly encrypts the votes in the order presented on the left side of the ballot, Vivian is allowed to get two ballots, one to audit and one to cast. For the audited ballot, Vivian can remove the scratch off surface and reveal the randomness that was used to produce the encryptions in the 2D bar code. Using this randomness, anyone can regenerate the 2D bar code and see if it is in accord with the order in which the candidates appear on the left side. In this case, the probability that Vivian does not detect her incorrectly formed ballot is 50%. Therefore the requirement for a "Presented ballots are well formed" check is satisfied.

Cast ballots are well-formed The cast ballot is considered to be the portion of the 2D bar code that corresponds to the marks made by the voter. To understand how Scratch&Vote satisfies the"Cast ballots are well formed" requirement we have to mention that the cast ballots are encrypted and are tallied in a homomorphic manner. All the encrypted ballots are aggregated (multiplied) and the aggregate is decrypted using distributed Elgamal decryption.

At the time when Vivian checks that her ballot is cor-

rectly formed, she can also check that the 2D bar code does not contain over-votes or negative votes. However, the "Cast ballots are well formed" requires that Ann be able to perform this check, since Vivian can collude with the voting system, as explained in the following potential attack. The system may hand Vivian a ballot that is known to have a large number of votes for the candidate Vivian supports, so Vivian has no interest in disclosing this. Vivian happily casts a ballot that contains a large number of votes for a candidate and a number of negative votes for another candidate. To make sure this does not happen, the system needs to prove to Ann that all the encrypted votes are well-formed. In practice this is done using known zero-knowledge proof techniques on the published encryptions. In such techniques the probability that the cast ballot is malformed and not detected is negligible. Therefore the requirement for a "Cast ballots are well formed" check is satisfied.

Tallied as recorded The votes are tallied in a homomorphic manner: after the encrypted votes are selected from the bar codes on each ballot, they are multiplied, resulting in an encryption of the entire tally. This encrypted tally can be decrypted in a publicly verifiable manner using a threshold scheme. In this case c_n is fixed and is very low, i.e. lower than 2^{-112}, which in turn means that the probability of cheating on decrypting the tally and not getting detected by the public is essentially zero. Therefore the requirement for a "Tallied as recorded" check is satisfied.

Each recorded ballot is subject to the "recorded as cast" check If two voters are given the same ballot, it may happen that they choose different candidates thus Ann can detect that the same receipt is posted twice on the bulletin board, with different selections.

In conclusion, the elections which use Scratch&Vote are end-to-end verifiable.

11.4 ThreeBallot

The ThreeBallot ballot contains four parts. The leftmost part has a list of candidates in canonical order (same order on all ballots). The next three parts are identical: each part has a place where the voter can put a mark for a candidate and a unique identifier at the bottom. Each mark for a candidate is a vote for that candidate. The trick is that the voter votes once for each candidate and an extra time for her favorite candidate. A ballot with an actual vote for Carol contains two votes for Carol and one vote for everyone else.

After the ballot is filled in by the voter, it is placed in a checker that verifies that it was correctly filled in (this is not trivial since there are many ways to incorrectly fill in a ThreeBallot). Then, the voter separates the three parts on the right from each other and chooses one at random to make a copy of. The copy becomes the voter's receipt. The left part with the names of the candidates is irrelevant, since it contains public information that is the same on all ballots. The three parts are deposited into a ballot box and counted as separate ballots. The counting results in each candidate having "v" votes above the number of votes cast, where "v" is the number of voters. This number is subtracted from the reported totals to find the count.

Presented ballots are well-formed In ThreeBallot, checking for correct printing is trivial, since initially all the ballots look the same (except for the unique serial numbers) and the filled-in ballots are not transformed in any way to produce the tally.

Cast ballots are well-formed ThreeBallot does not satisfy the requirement for a "Cast ballots are well formed" check. The reason is that Ann cannot verify that Vivian did not make three marks for her favorite candidate and no marks for another candidate, which would result in an extra (invalid) vote for her favorite candidate and a negative vote for the other one. While Vivian can make this check and the ThreeBallot system can make this check, Vivian may have an interest in marking her ballot in this manner and she may also be colluding with the ThreeBallot's checker.

All the ballots that constitute a voted ThreeBallot ballot ($3\times$v of them) are posted on a public bulletin board. Using the unique identifier at the bottom of her receipt, the voter can check that the copy that she kept (of one of the three ballots) is correctly published on the bulletin board. She can complain with a valid proof if it does not. Since the voting system does not know which of the three parts the voter kept, if it tries to modify a ballot, it would have to guess which parts the voter did not keep. For every ballot, the system has a 66% chance of correctly guessing which part the voter did not keep. Therefore Vivian can probabilistically check that her entire ballot is correctly posted. If Vivian does notice that her receipt is incorrectly posted, she can present her receipt as irrefutable proof of malfeasance. Therefore the requirement for a "Recorded as cast" check is satisfied.

Tallied as recorded The requirement for a "Tallied as recorded" check is trivially satisfied, because the bulletin board contains clear-text ballots that are tallied by Ann. The probability that cheating occurs and is not detected is zero. Not many systems offer this level of assurance.

Each recorded ballot is subject to the "recorded as cast" check If two voters are given the same ballot, it may happen that they mark their ballots differently (regardless if they vote for the same candidate or not) and thus Ann can detect that the same receipt is posted twice on the bulletin board, with different selections. However, if Vivian indicates to a machine her favorite candidate and the machine produces the ThreeBallot, then the ma-

chine may give both Vivian and Victor the same ballot and Ann is not able to detect that each recorded ballot is subject to the "recorded as cast" check, thus this check fails. Note that, in general, the machine can reuses single parts of ThreeBallots to construct whatever vote desired This is a generalization of the attack on only two voters.

Follows the Protocol The way ThreeBallot is described [11], the voter indicates to the checker which ballot she wants to keep as a receipt. The checker may ignore the voter's choice and give her another ballot; the voter knows, but cannot prove, that her ballot selection was ignored. However, there may be some mechanical designs (e.g. using levers) which would allow the voter to prove that her choice was ignored, and thus satisfy the "voting system follows protocol" requirement.

Note that there is no mention of cryptography in how ThreeBallot works.

Because Ann is not able to check if "Cast ballots are well formed", ThreeBallot does not qualify as an end-to-end verifiable system.

11.5 Scantegrity II

A Scantegrity II [6] ballot looks like a regular optical scan ballot, but has symbols that are printed inside the ovals, in invisible ink. The ballot is marked with a special pen that darkens the oval and reveals the code at the same time. The voter can write down the code to produce a knowledge-based receipt. All receipts are published on a public bulletin board. To produce the tally, Scantegrity II uses a two-mix construct that is simpler than the one in PunchScan, but offers essentially the same integrity guarantees. It is also based on commitments.

Presented ballots are well-formed To check that her ballot is well formed, Vivian can choose a number of ballots, one to vote and the rest to audit, just like in Prêt à Voter or Scratch&Vote. The same discussion applies.

Cast ballots are well-formed The Scantegrity II cast ballot cannot contain any negative votes, and over-votes can be detected by anyone who inspects the receipts published on the public bulletin board. Therefore the requirement for a "Cast ballots are well formed" check is satisfied.

Tallied as recorded Scantegrity II uses a publicly verifiable two-mix like mechanism to check that all the cast ballots have been "Tallied as recorded". Typical values for c_n are either $c_n = \frac{1}{2^n}$ if randomized partial checking per ballot is used, or $c_n = \frac{1}{2^d}$, where d is a fixed parameter if pairs of mixes are partially checked. Therefore the requirement for a "Tallied as recorded" check is satisfied

Recorded as cast There are some unique characteristics of the "Recorded as cast" check for Scantegrity. Vivian can check that her receipt is correctly posted on the public bulletin board. If Vivian notices that her receipt is incorrectly posted, she can bring as evidence the confirmation code that she knows. Assume that Vivian wants to cast doubt on the election and claim her receipt is incorrectly posted when in fact it is correctly posted. Then Vivian has to correctly guess a confirmation code. The probability that Vivian guesses a valid confirmation code which is different than the one she legitimately received depends on how difficult the confirmation codes are to guess, i.e. on how long they are. Therefore the proof that Vivian brings when she notices that her receipt is incorrectly posted is probabilistic (not irrefutable as in all other examples).

The first description of Scantegrity II does not conform to the "Recorded as cast" requirement, because the voting system can add a mark to the ballot cast by Vivian. For example, if Vivian does not make any selection for a race she does not get any confirmation code. After the ballot is cast, Scantegrity II marks Vivian's ballot and publishes a confirmation code. Vivian cannot prove that she does not know the posted confirmation code (especially since it is publicly posted), nor that she did not cause that code to appear on the ballot. Another scenario is when Vivian votes for a candidate, gets a confirmation code, but the Scantegrity II system adds a second mark to that race, causing it to be over-voted and thus invalid. Both confirmation codes are published on the bulletin board and again Vivian cannot prove that she does not know one of the codes or that she was not the one making it.

In the scenarios above, Vivian is able to detect that her ballot is not recorded as cast, but she cannot prove it. This would satisfy the weak version of the requirement for a check, but not the strong version.

Each recorded ballot is subject to the "recorded as cast" check If two voters are given the same ballot, it may happen that they choose different candidates thus Ann can detect that the same receipt is posted twice on the bulletin board, with different selections. However, because the "recorded as cast" check is weak, a single receipt can be posted with the confirmation codes for both candidates chosen by the two voters.

In conclusion, the elections which use Scantegrity II are end-to-end verifiable with a weak "recorded as cast" check.

11.6 Helios

Helios [1] is the name of the implementation of the more general Benaloh challenge [4] voting system. The main idea of the Benaloh challenge is that Vivian is presented with an encryption of her vote, and she is allowed to choose between decrypting it or casting the encrypted ballot. When Vivian chooses to decrypt her ballot, she is able to check that the choices inside the encryption are

correct. She is then allowed to repeat the process until she chooses to cast a ballot (without choosing to decrypt it). In Helios, the encrypted files are tabulated in a homomorphic manner, similar to Scratch&Vote.

Presented ballots are well-formed To check that the ballot to be cast by Vivian is well formed, Vivian may ask Helios to fully open her ballot and check that the choices inside the encryption are hers. If Vivian notices that the choices inside the encryption are different from the choices she indicated, then she knows that Helios tried to change her choices, but she does not have proof that the choices inside the now open encryption are different from the choices she indicated to Helios. Thus Helios complies with the weak version of the requirement for a "Cast ballots are well formed" check.

Cast ballots are well-formed When Vivian asks Helios to open her ballot, she can check that "cast ballot is well formed", i.e. that the ballot does not contain any over-votes or negative votes. But Ann is the one who needs to be able to perform this check, thus, for each cast ballot, Helios needs to publish a proof that the cast ballot is well formed. In practice this can be done using known zero-knowledge proof techniques, for which the probability that the cast ballot is malformed and not detected is lower than, e.g. 2^{-112}. Therefore the requirement for a "Cast ballots are well formed" check is satisfied.

Tallied as recorded The votes are tallied in a homomorphic manner: after the encrypted votes are cast they are multiplied together, resulting in an encryption of the entire tally. This encrypted tally can be decrypted in a publicly verifiable manner using a threshold scheme. In this case c_n is fixed and is very low, i.e. lower than 2^{-112}, which in turn means that the probability of cheating on decrypting the tally and not getting detected by the public is essentially zero. Therefore the requirement for a "Tallied as recorded" check is satisfied.

Follows the Protocol Since Helios uses a remote electronic interface to interact with the voter, the system may choose to disobey the voters commands. Vivian may ask Helios to open the encrypted vote, but the system may ignore this input and cast the ballot. Vivian does not have proof that the system was disobedient. Thus the requirement for a "voting system follows the protocol" check is weak for the remote version of Helios. Note that the underlying Benaloh challenge system implemented in a polling place may satisfy the requirement.

Each recorded ballot is subject to the "recorded as cast" check In Helios, the system knows the selections of the voters before it gives them a receipt, and thus can choose to give the same receipt to two voters that voted the same way. However, Vivian may choose to audit her ballot and Victor may choose to cast it and thus Ann can detect that the same receipt is posted twice on the bulletin board, with different states: one is marked as cast and one as audited.

In conclusion, the elections which use Helios are end-to-end verifiable, but satisfy a weak version of the requirement for a "Cast ballots are well formed" check.

12 Conclusions

We presented precise performance requirements for end-to-end verifiable elections. We identified six properties which are required to hold for voting system designs which can be used in end-to-end verifiable elections: 1. The voter is able to check that her ballot represents a vote for the candidate she intended. 2. Anyone is able to check that valid ballots do not contain over-votes or negative votes. 3. The voter can check that her ballot is recorded as she cast it. 4. Anyone is able to check that all the recorded ballots have been correctly tallied. 5. Anyone is able to check that voters and the general public has the same view of the election records. 6. Anyone can check that any cast ballot has a voter that can perform check number three. In addition, the voting system cannot ignore a voter's choice.

For each requirement, we specify *who* can check it, *when* it can be checked, *what* exactly is checked and what kind of *proof* exists to demonstrate that the system failed the check, if it did. We briefly discuss how six previously proposed voting systems comply or not with the stated performance requirements (see Table 1).

Note that we do not present design requirements; we do not specify general steps necessary for a system to be built, but rather what properties we wish to achieve. The requirements do not mention receipts, cryptography or the use of a bulletin board. Also, the requirements are only pertinent to integrity. Separate documents should address aspects that are equally important to an election: privacy, coercion, authentication, usability, accessibility, scaling, cost, etc.

Our aim is to give a precise meaning to the term *end-to-end verifiable elections*. We hope that future designs will describe how they comply with these performance requirements, such that the term is interpreted consistently by various approaches.

References

[1] Ben Adida. Helios: Web-based open audit voting. In *Proceedings of the Fourteenth USENIX Security Symposium (USENIX Security 2008)*. Usenix, July 2008.

[2] Ben Adida and Ronald L. Rivest. Scratch & Vote: self-contained paper-based cryptographic voting. In *WPES '06: Proceedings of the 5th ACM workshop on Privacy in electronic society*, pages 29–40, New York, NY, USA, 2006. ACM Press.

[3] Roberto Arajo, Ricardo Felipe Custdio, and Jeroen van de Graaf;. A verifiable voting protocol based on farnel. In *IAVoSS Workshop On Trustworthy Elections (WOTE 2007)*, University of Ottawa, Canada, June 2007.

	BaWF	CBaWF	RaC	TaR	C	ERBiSttRaCC	FtP
Prêt à Voter	✓	✓	✓	✓	✓	✓	✓
PunchScan	✓	✓	✓	✓	✓	✓	✓
Scratch&Vote	✓	✓	✓	✓	✓	✓	✓
ThreeBallot	✓	no	✓	✓	✓	✓	✓
Scantegrity II	✓	✓	weak for added votes	✓	✓	✓	✓
Helios	weak	✓	✓	✓	✓	✓	weak

Table 1: Some voting systems and their conformance to the proposed end-to-end performance requirements. BaWF=Ballots are Well Formed; CBaWF=Cast BaWF; RaC=Recorded as Cast; TaR=Tallied as Recorded; C=Consistency; ERBiSttRaCC=Each recorded ballot is subject to the "recorded as cast" check; FtP= Follows the Protocol

[4] Josh Benaloh. Administrative and public verifiability: Can we have both? In *IAVoSS Workshop On Trustworthy Elections (WOTE 2008)*, KU Leuven, Belgium, July 2008.

[5] David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, pages 38–47, January/February 2004.

[6] David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, and Alan T. Sherman. Scantegrity ii: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes. In *EVT'08: Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop*. USENIX Association, 2008.

[7] David Chaum, Peter Y. A. Ryan, and Steve Schneider. A practical voter-verifiable election scheme. In *In Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann, editors, ESORICS, volume 3679 of Lecture Notes in Computer Science*, pages 118–139. Springer, 2005.

[8] Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *Proceedings of the 11th USENIX Security Symposium*, pages 339–353, Berkeley, CA, USA, 2002. USENIX Association.

[9] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *8th ACM Conference on Computer and Communications Security*, pages 116–125, 2001.

[10] Stefan Popoveniuc and Ben Hosp. An introduction to PunchScan. In *IAVoSS Workshop On Trustworthy Elections (WOTE 2006)*, Robinson College, Cambridge UK, June 2006.

[11] Ronald L. Rivest and Warren D. Smith. Three voting protocols: ThreeBallot, VAV, and Twin. In *EVT'07: Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology*, pages 16–16, Berkeley, CA, USA, 2007. USENIX Association.

Notes

[1]For example, election law will likely come into play in deciding how to deal with failed checks; if those checks are routinely ignored, even a wonderful end-to-end voting system won't necessarily improve the integrity of the election.

[2]Note that corrupt election officials can insert votes in the names of nonexistent or absent voters in any voting system; such attacks are outside the scope of this paper.

[3]Keywords in the skeleton should be ignored while reading a requirement. The bolded keywords are present for structural reasons and should not impede the fluent reading of a requirement. For example, the first requirement should be read as: If the ballot to be cast by Vivian is not well-formed, then, at any time after the election, Vivian is able to detect if the vote she is about to cast does not represent a vote for the candidate(s) she intended.

[4]We realize that, unfortunately, Farnel[3] does not fit this requirement, since Vivian is not checking her ballot. It would be possible to re-write this requirement for the collection of ballots rather than individual ballots or to allow one voter to verify the properties associated with other voters' ballots.

[5]Clearly, the probability should be high enough that election fraud is likely to be detected; unfortunately we do not currently know what the right lower-limit on this probability needs to be for all elections everywhere.

[6]If Vivian is able to falsely prove, with non-negligible probability, that the voting system did not record her ballot correctly, she may cause a denial of service attack. As we restrict our focus to integrity, we do not elaborate on the different ways in which denial of service attacks can be conducted. Note that Scantegrity II[6] is the only currently proposed voting system of which we're aware in which Vivian may produce a false receipt which seems valid with non-negligible probability.

[7]We recognize that this requirement makes our definition less powerful, but there appears to be no way to specify *a priori* all the different checks of this kind that might be needed by every possible end-to-end voting system used in any election.

[8]This poses some problems for "something you know" type receipts. Scantegrity II overcomes this problem by providing a "none of the above" choice in all races, and by having the election official provide the voter with an additional code if the ballot is fully marked, as determined by the scanner

[9]That beacon then becomes a part of the election which must follow the protocol, as required by our generic requirement, or be detected not doing so with significant probability.

[10]It seems unreasonable to make such a weak requirement on the probability of detecting an error, but we simply do not know what the right limits on this probability should be for all elections everywhere.

[11]NIST does not endorse any of the examples presented