Towards Secure Quadratic Voting

Sunoo Park
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139
sunoo@mit.edu

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

Abstract

We provide an overview of some of the security issues involved in securely implementing Lalley and Weyl's "Quadratic Voting" [33], and suggest some possible implementation architectures. Our proposals blend *end-to-end verifiable* voting methods with anonymous payments. We also consider new refund rules for quadratic voting, such as a "lottery" method.

1 Introduction

The problem of reliably implementing an election is ages-old, and has been recognized at least since the use of the secret ballot in Ancient Greece and Rome. An ideal voting system has a complex set of desiderata, and must uphold them in a threat model where no one can be trusted: in general, neither voters, nor the election authorities, nor the equipment used in the election can be considered trustworthy by everyone with a stake in the outcome.

Four important properties of a voting system are *verifiability*, *secrecy*, *robustness against false accusations*, and *usability*. An ideal voting system should produce a trail of evidence such that any observer can *verify* the accuracy of the reported election outcome based on the evidence. Also, the system should be *robust against false accusations*, meaning that if the election is indeed conducted correctly, then it is not possible to produce (false) evidence that the election was not conducted correctly.[1] Moreover, the vote of any individual should be *secret*:

[1] While this is an already-existing concept in the secure voting literature, it does not have a widely agreed-upon name. As far as we know, this is the first use of the term "robustness against false accusations" to describe this property.

this is an essential defense against vote selling and coercion. Finally, the system should be able to be understood and *used* by all eligible voters.

These goals do not always mesh well together: indeed, the first two often seem to be in direct opposition to each other. Researchers have studied election integrity for decades, and numerous systems have been proposed, which draw upon diverse mechanisms from cryptography, statistics, and systems security and offer varying guarantees on verifiability and secrecy.

1.1 Quadratic voting

Lalley and Weyl's "Quadratic Voting" (QV) [33] is a voting scheme for electing one out of two candidates,[2] which departs from the concept of "one person, one vote" and instead allows each eligible voter to cast multiple votes for any single candidate, thereby expressing the magnitude of his or her preference for the selected candidate.[3] Naturally, there is an associated cost (otherwise, everyone would be incentivized to cast arbitrarily many votes for their candidate): voters must pay to cast votes, and the pricing increases *quadratically*. That is, if buying one vote costs 1 cent, then buying two votes costs 4 cents, buying three votes costs 9 cents, and so on. After the election, the total revenue is reallocated among the voters according to a *refund rule*: concretely, Lalley and Weyl suggest that each voter get an equal share of the election revenue.

An attractive feature of quadratic voting is that it attains *efficiency* (in the economic sense: a more efficient system has less potential to *improve* the outcome according to a certain measure, i.e., it is closer to optimal by that measure). Namely, [33] shows that quadratic voting achieves optimal efficiency in the sense that asympotically, the system's utilitarian inefficiency tends to zero. It turns out that quadratic voting also achieves *Pareto efficiency* (again, asympotically), which means that nobody can be made better off without making someone else worse off. The efficiency of QV stems from the quadratic pricing, which Lalley and Weyl show incentivizes voters to cast a number of votes which honestly reflects how much they care about the issues being voted on. Lalley and Weyl envision quadratic voting to be a potentially advantageous decision-making mechanism for diverse settings; they give examples ranging from market surveys to corporate governance to national political elections. We refer to their paper [33] for further discussion and motivation.

Quadratic voting is interesting from a security viewpoint because it requires integration of secure *voting* and *payments*: two functionalities that are each quite challenging to implement securely. In this paper, we provide an overview of some of the security issues involved in implementing quadratic voting, and propose some potential implementation architectures.

[2] All expositions of QV of which we are aware focus solely on the two-candidate case. Extending to many candidates may well be possible, but generalizing the analysis of [33] to many candidates seems possibly non-trivial and the details have not been worked through. In this work, we consider only the better-studied, two-candidate case.

[3] A somewhat similar voting scheme is *range voting* [43]. In range voting, votes can have different magnitudes. Unlike QV, however, range voting does not assign a cost per vote (and is mainly considered useful in the many-candidate setting).

Collusion Quadratic voting as presented in [33] is inherently susceptible to collusion between the voters: a group of voters supporting the same candidate who pool their money evenly amongst themselves will be able to effectively buy votes at a cheaper rate than voters who do not collude. This means that voters who have strong preferences will be incentivized to pay their friends (and perhaps also strangers) to cast votes for their favored candidate.

This issue is pointed out but unaddressed by [33]. A more recent work by Weyl [52] analyzes the robustness of QV against variations in voter behavior, including collusion, but does not eliminate the problem.

Within the scope of this work, we study how to securely implement the quadratic voting scheme as presented in [33], and do not address the collusion problem. We remark, however, that collusion is a real potential security concern which we think it would be important to address in a deployment of QV in any high-stakes setting where adversarial behavior by voters is possible or likely.

2 Background on secure voting and payments

2.1 Secure voting

Voting is, of course, central to democracy.

Yet implementing voting securely is challenging, for many reasons. First, the secret ballot makes typical security methods inapplicable. For example, the system cannot give a voter a receipt showing how she voted, lest she use that to sell her vote. Second, in a typical election there is really no "trusted party"; the integrity of the election depends on having many eyes on each step of the process. Third, the function of an election is more demanding than a naive observer might expect. A good election system must not only produce the correct outcome, but also provide convincing *evidence* that the outcome is correct, sufficient to convince a skeptical losing candidate that he lost fair and square. These requirements are in tension with each other; it is difficult (but not impossible) to satisfy them all simultaneously.

There is a large literature on election methods and technology. Jones and Simons [30] provide an overview of recent attempts at applying technological innovation to voting, not always successfully.

Many recent proposals fall within the camp of "end-to-end verifiable (E2E) voting". In an E2E method, verifiability is provided for each of the three basic properties: "cast as intended", "collected as cast", and "tallied as collected". The Overseas Vote Foundation recently issued a report [23] that surveys end-to-end verifiable voting technology and its potential applicability to voting over the internet. Two notable end-to-end verifiable voting implementation projects are Scantegrity [18] and STAR-Vote [7]. Another one is Helios [1], which is especially designed for remote voting over the internet in low-stakes settings.[4]

[4] Adida, the author of Helios, discourages the use of Helios for high-stakes elections since remote voting using currently available technology is inherently risky in scenarios where strongly adversarial behavior such as coercion and/or vote-buying are likely to occur [1].

In this paper we follow this trend, and describe E2E voting methods for quadratic voting that are modelled after the STAR-Vote design (in the polling-site setting) and the Helios design (for remote voting in low-stakes settings).

2.2 Secure payments

"Secure payments" is an umbrella term that spans a vast range of concepts, from cash (e.g., preventing counterfeits) to traditional banking (e.g., payments between financial institutions) to the more recent technologies underlying credit cards (e.g., chip and PIN), online purchases, and cryptocurrencies (e.g., Bitcoin). Within the scope of this paper, we are interested in only a select few aspects of payments, which are directly relevant to constructing a verifiable and secret quadratic voting system. The properties that we require of the payment scheme are discussed in Section 3.1.

3 Implementing quadratic voting

Lalley and Weyl envision quadratic voting to be a useful decision-making mechanism for diverse settings, from small-scale polls and surveys to potentially nationwide elections. These different settings have very different security requirements. In this paper, we consider two broad categories of collective decision-making scenarios.

1. **Elections**, where eligibility to vote is tied to a "*real-world*" identity (such as a presidential election, where eligibility is determined by citizenship). For elections, we consider the "polling place" setting, where participants must come to a designated physical venue in order to vote. In this setting, we assume that payments are made in cash.

2. **Surveys**, where eligibility is linked to a *virtual* identity (e.g. an email address). In this setting, eligibility to vote can be checked digitally online, and the stakes of the outcome tend to be lower, so we consider a fully online voting process. As we shall discuss, a main interesting challenge in this case is to implement electronic payments in such a way that the payment amounts cannot be traced back to individual identities.

At a high level, the steps involved in a secure implementation of quadratic voting are as follows.

1. *Identity check:* Only voters who can demonstrate their eligibility should be allowed to vote.

2. *Create a vote:* Eligible voters should have a method of submitting payment for and creating a ballot indicating their desired number of votes for their chosen candidate.

3. *Cast a vote:* Completed ballots should be submitted to the election authority. Only ballots created by eligible voters who have fully paid for their votes are considered to be valid submissions.

4. *Tally votes:* After all voters have submitted their votes, the election authority should be able to determine the outcome of the election and announce it.

5. *Sum payments:* The election authority should be able to determine the total amount of money paid for votes in the election, and announce it.

6. *Audit:* An auditing procedure is undertaken by voters and/or election officials and/or third parties to check that the election was conducted properly and the outcome and payment total were correctly computed.

7. *Reimbursements:* Each person who voted in the election should be paid the appropriate refund from the election revenue.

3.1 Verifiability, robustness against false accusations, and secrecy for QV

Each of the *verifiability*, *robustness against false accusations*, and *secrecy* requirements for traditional voting schemes need to be augmented with additional requirements for quadratic voting, due to the payments aspect.

3.1.1 Verifiability

The traditional requirement of verifiability is that the outcome of an election should be able to be verified by any independent observer. The natural extension of verifiability to the context of quadratic voting requires additionally that the total election revenue (and the fact that the revenue was redistributed to the voters as required) should be able to be verified by any independent observer. Note that the announced outcome of the election may be just the winner, or (more usually) the total number of votes for each option.

3.1.2 Robustness against false accusations

The usual requirement of robustness against false accusations is that if the election was indeed conducted correctly, no party should be able to produce false evidence that any voter's vote was incorrectly counted. For QV, this requirement is augmented to also require that no party can produce false evidence that any voter was incorrectly reimbursed.

3.1.3 Secrecy

Secrecy requires that no individual can convince someone else (beyond doubt) of the vote he cast. There are some situations where this is unavoidable: for example, in the case of a unanimous vote, it can be inferred which candidate

any individual voter voted for. Therefore, we technically require that no individual can convince someone of the vote he/she cast, *beyond what can already be inferred from the publicly announced outcome of the election*: i.e., from the final vote tally and the total amount of election revenue. This requirement should be satisfied even if the voter herself is *trying* to convince someone truthfully of the vote she cast: this is important in order to avoid coercion and vote-buying.

In traditional voting systems, the secret information is just *who/what the voter voted for*. The natural extension of secrecy to quadratic voting requires that the *magnitude of any individual's vote* must additionally be kept secret. In fact, to prevent coercion and vote-buying in QV, we require an even stronger condition: that even the set of magnitudes of all votes cast must not be revealed.

To see why this requirement is needed, consider a coercer who tells a voter to vote a specific number of votes (say, \$42.13 worth) for a particular candidate. The coercer will "check" whether the voter has complied with his demand, by checking the list of amounts voted to make sure the amount \$42.13 is there. As long as there is a good chance that nobody else will vote that exact amount, then the voter will be compelled to vote as the coercer says, since non-compliance is reasonably likely to be detected by the coercer. We remark that this type of attack is related to the *pattern voting* attack which has been discussed in the standard voting literature [42].

Note that the coercer cannot force a voter to vote for a particular candidate using this method, as the list does not reveal which candidate was voted for. Nonetheless, if the coercer chooses voters whose preference he knows matches his own (but would vote less than \$42.13 if not coerced), then he can be reasonably sure that the voters indeed voted for the correct candidate. This means that for example, if the coercer successfully coerces two voters in this way – and even if he fully reimburses the \$42.13 incurred by each – he has effectively bought $2\sqrt{4213}$ votes at the price of \$84.26, rather than $(2\sqrt{4213})^2 = \$168.52$ as it should rightfully cost by quadratic pricing.

4 Implementation of in-person voting

In this section, we propose an implementation of quadratic voting where eligibility to vote is associated with a real-world identity. We remark that we assume a payment system which is discrete-valued, like every existing currency, whereas Lalley and Weyl's analysis models money as continuous (real-valued). Provided that the discrete-valued payments have sufficiently fine granularity, this should be a reasonable approximation to the real-valued case.

In this work, we refer to currency in dollars and cents, and assume that one vote costs one cent. We write n to denote the total number of voters who vote in an election (note that this is not necessarily equal to the number of eligible voters), and $[n] = \{1, \ldots, n\}$ denotes the set of all voters who voted.

4.1 Physical implementation with wax

Consider the following toy implementation of quadratic voting, which uses quantities of wax to represent votes: wax has the useful physical property that chunks of it can be melted together and this irreversibly deletes all information about the sizes of the original chunks.

1. *Identity check:* A voter (call her Alice) walks into the polling place and presents her ID document to a designated poll worker (call him Bob). Bob has a list of the names and addresses of all voters who are eligible to vote at this polling place. If he verifies that Alice's ID document is valid, then Bob crosses Alice's name off the list and gives Alice a "voter token"[5] which indicates that she is authorized to vote.

2. *Create a vote:* Alice takes her token and approaches a different poll worker (call him Charlie) who is in charge of creating votes. Alice gives Charlie her voter token, tells him the number v of votes she would like to buy, and gives Charlie v^2 in cash. Charlie places the cash in a lock-box[6] (while Alice watches) and gives Alice an embossed wax block[7] of mass v.

3. *Cast a vote:* Alice enters a curtained voting area where there are two large opaque containers, each labeled with one candidate's name. Each container has an opening on the top[8]. To place her vote for a candidate, Alice drops her wax block into the corresponding container.

4. *Tally votes:* In order to count the votes, the total mass of wax in each container is weighed. Before the containers are opened, they must be heated so that the wax melts and becomes homogeneous.

5. *Sum payments:* In order to sum the payments, the lock-box is opened and the cash is counted. The opening and counting process should be closely monitored by having many people present, and perhaps also being filmed and shown on television.

6. *Audit:* In this toy example, there is no separate auditing step: its purpose is served by the public monitoring of the tallying process for votes and payments, described above.

[5] The voter token should be hard to forge; e.g. it could be a special kind of coin minted specially for this election.

[6] The lock-box should be thought of as a "piggy bank" with a small opening to insert money, from which it is physically impossible to remove money unless the box is unlocked.

[7] The wax block should be embossed with a hard-to-forge pattern specific to this election. Also, note that the wax need not be a monolithic block. For practical purposes, since we assume discrete money, it would likely be more manageable to have wax "coins" in set denominations, e.g. a 1-gram coin, 4-gram coin, 9-gram coin, etc.

[8] In order to ensure that only valid wax blocks can enter the containers, the opening of the container should actually be protected by a box which mechanically checks whether the wax block has the correct embossing, and only drops it into the container if the check is passed.

7. *Reimbursements:* Each person who voted in the election (i.e. each person who is crossed off of Bob's list) is mailed a check for the amount of V/n where V is the election revenue and n is the total number of voters.

Verifiability and robustness against false accusations The above-described voting scheme's verifiability and robustness against false accusations follow from the public monitoring of the tallying process for votes and payments, and the fact that from the total amount of election revenue, each voter can check that his/her reimbursement check is for the correct amount. (We assume that voters cannot forge reimbursement checks for incorrect amounts.)

Secrecy The secrecy of this voting scheme follows from:

1. The special physical property of wax that we highlighted at the beginning of the section: namely, that once the wax chunks are melted together in the "Tally votes" step, the sizes of the wax chunks are irrecoverably lost, and thus only the total mass can be known since the wax containers are opened only after the melting occurs.
2. The opacity and physical security of the wax containers and the lock-box, i.e., the facts that
   (a) nobody can see inside them before they are opened; and
   (b) a voter who physically places his/her wax block into the container, or watches the poll worker physically place an amount of cash into the lock-box, can be sure that the wax block (or cash) indeed entered the container and will stay there until the container is opened; and
   (c) cash is anonymous, so when the lock-box is opened, an observer cannot tell what are the amounts contributed by any individual voters.

4.2 Cryptographic tools

In order to securely implement quadratic voting using cryptographic (rather than physical, wax-based) guarantees, we need to introduce some cryptographic tools. The descriptions we give here are relatively informal; for more detailed, rigorous definitions, we suggest consulting any standard textbook on cryptography (such as "Introduction to Modern Cryptography" by Katz and Lindell [31]).

4.2.1 Encryption

Secret-key encryption The simplest type of encryption is *secret-key encryption*. A *secret-key encryption scheme* SKE consists of three algorithms: $\mathsf{SKE.Gen}$, $\mathsf{SKE.Enc}$, and $\mathsf{SKE.Dec}$.

- The *key generation algorithm* $\mathsf{SKE.Gen}$ takes as input a "security parameter" indicating the level of security desired, and outputs a *secret key* k.

- The *encryption algorithm* $\mathsf{SKE.Enc}$, when given a *secret key* k and a message m as input, outputs a *ciphertext* c. To someone who does not know the secret key k, the ciphertext c reveals no information about the underlying message m.
- The *decryption algorithm* $\mathsf{SKE.Dec}$ allows an authorized recipient who does know the secret key to recover the message: on input k and c, $\mathsf{SKE.Dec}$ outputs the original message m.

Public-key encryption A more advanced type of encryption is *public-key encryption*. A *public-key encryption scheme* PKE also consists of three algorithms: $\mathsf{PKE.Gen}$, $\mathsf{PKE.Enc}$, and $\mathsf{PKE.Dec}$.

- The *key generation algorithm* $\mathsf{PKE.Gen}$ takes as input a "security parameter" indicating the level of security desired, and outputs a pair of keys (k, K). As before, k is the secret key; and K is a *public key* which can be made publicly known.
- The *encryption algorithm* $\mathsf{PKE.Enc}$, when given a *public key* K and a message m as input, outputs a *ciphertext* c. To someone who does not know the secret key k, the ciphertext c reveals no information about the underlying message m.
- The *decryption algorithm* $\mathsf{PKE.Dec}$ allows an authorized recipient who does know the secret key to recover the message: on input k and c, $\mathsf{PKE.Dec}$ outputs the original message m.

In this work, we are only interested in *public-key* encryption.

Note that an important feature of secure encryption schemes is that the encryption algorithm is *randomized*. This means that two encryptions of the same message are not the same.[9] The importance of randomized encryption is well illustrated by settings such as voting, where there are only a few candidates to choose from: if two encryptions of the same message resulted in the same ciphertext, then an attacker could figure out people's votes by just running the encryption algorithm on all the possible options, and comparing!

4.2.2 Homomorphic encryption

Additively homomorphic encryption is a special type of encryption which allows someone who is given two ciphertexts

$$c = \mathsf{PKE.Enc}(K, m) \text{ and } c' = \mathsf{PKE.Enc}(K, m'),$$

but does *not* know the secret key k or the messages m and m', to compute a new ciphertext c'' which encrypts the sum $m + m'$ of the two original messages. That is, $\mathsf{PKE.Dec}(k, c'') = m + m'$. An example of a well-known public-key

[9] In fact, the formal security requirement says that two encryptions of the same message must be *indistinguishable* from two encryptions of totally different messages.

encryption scheme which has this property is the Paillier scheme, which has been considered for voting applications for exactly this reason [5, 38, 20].[10]

In the context of voting, this is a very useful feature, since it means that even if individual votes are encrypted, it is possible to compute a ciphertext which encrypts the sum of all votes *without decrypting the individual votes*. This is important to preserve the secrecy of individuals' votes while still ensuring that the election outcome is correctly computed.

For quadratic voting, we would like something slightly more than additively homomorphic encryption: namely, in addition, to adding up the votes $v_1, \ldots, v_n$ of the voters in order to compute the election outcome, we would like to add up the payments $v_1^2, \ldots, v_n^2$ of the voters in order to compute the election revenue. In the terminology of the encryption scheme, it would suffice if someone who is given two ciphertexts

$$c = \mathsf{PKE.Enc}(K, m) \text{ and } c' = \mathsf{PKE.Enc}(K, m'),$$

but does *not* know the secret key k or the messages m and m', has the ability to compute a new ciphertext c'' which encrypts the product $m \times m'$ of the two original messages. With such an encryption scheme, if we are given encryptions

$$c_1 = \mathsf{PKE.Enc}(K, v_1), \ldots, c_n = \mathsf{PKE.Enc}(K, v_n)$$

of the votes $v_1, \ldots, v_n$, we can use the ciphertext multiplication procedure to compute $c'_1, \ldots, c'_n$ which are encryptions of $v_1^2, \ldots, v_n^2$ respectively, then use the ciphertext addition procedure on the new ciphertexts $c'_1, \ldots, c'_n$ to compute a final ciphertext which encrypts the sum $\sum_{i \in [n]} v_i^2$ of payment amounts.

Fortunately, efficient encryption schemes with exactly this property are known: we suggest the Boneh-Goh-Nissim encryption scheme [12].[11] Note that the presentation we have given here is somewhat simplified; we refer the interested reader to the original paper [12] for full details of the specific suggested encryption scheme. More generally, we refer to [22, 50, 25, 36] for a survey of homomorphic encryption techniques and definitions.[12]

[10] The standard definition of additively homomorphic encryption actually requires that the described property holds for adding up not only *two* ciphertexts, but any (polynomial) number of ciphertexts. Note that in fact, we need this more general property in order to use additively homomorphic encryption for tallying (more than two) votes.

[11] We remark that in fact, encryption schemes with an even stronger property are known: *fully homomorphic encryption* schemes allow arbitrary multiplications and additions of ciphertexts, whereas the [12] scheme allows arbitrary additions but *only one multiplication per ciphertext*. However, the known constructions of fully homomorphic encryption are too inefficient to be practical. We suggest the [12] scheme since it has the required properties for QV tallying (only one multiplication per ciphertext is needed, for the squaring), and – unlike the fully homomorphic schemes – it is efficient enough for practical use.

[12] The first reference [22] is self-designated "for non-specialists", the next two [50, 25] are more recent and somewhat more technical surveys written by cryptographers at the forefront of theoretical developments in the field, and the last reference offers a perspective on practically deploying homomorphic encryption.

4.2.3 Threshold decryption

In the encryption schemes described so far, we assumed there was an authorized recipient in possession of the secret key, who would be able to decrypt ciphertexts. In some situations, it may be desirable that no individual entity has the power to decrypt ciphertexts alone. For example, there may be a *set* of people who have to collaborate in order to decrypt ciphertexts.

A *threshold decryption scheme* provides a method of *sharing* a decryption key among m parties, such that for some threshold $t \leq m$, any t of the parties together have the power to decrypt, but any subset of fewer than t parties cannot decrypt. The first construction of such a scheme was given by Shamir in 1979 [48]. We will use t such that $1 < t < m$.

Why $1 < t < n$? If $t = n$, then if even one of the people in the set is malicious or unavailable, the others will not be able to decrypt. We would like our scheme to be robust against such failures, so we set t lower than n. On the other hand, if t parties are collude maliciously, then they can decrypt ciphertexts (that they should not be able to decrypt): we set t reasonably large in order to reduce the probability of such a malicious collusion.

4.2.4 Digital signatures

For traditional handwritten signatures that are used to sign documents, there are three main desirable properties:

1. a signature should be unique to an individual person,
2. anyone should be able to look at a genuine signature on a document and be convinced of (a) its authenticity and (b) the fact that it is indeed a signature on *this particular document*,[13]
3. and yet it should be hard for other people to forge.

Digital signatures are the electronic equivalent of traditional handwritten signatures. A *digital signature scheme* consists of three algorithms: Gen, Sign, and Verify.

- The *key generation algorithm* Gen takes as input a "security parameter" indicating the level of security desired, and outputs a pair of keys (k, K). Here, k is the *signing key* which should be kept secret, and K is the *verification key* which can be made publicly known.
- The *signing algorithm* Sign, when given a *signing key* k and a message m as input, outputs a *signature* σ.

[13] In the case of handwritten signatures, the latter property is effectively enforced by having the signature physically bound to the document, i.e., on the same piece of paper.

- The *verification algorithm* Verify allows anyone who knows the public verification key K to check the validity of the signature: on input K, m, and σ, Verify outputs 1 if σ is indeed a signature on m produced using k, and outputs 0 otherwise.

An important difference between digital signatures and handwritten signatures is that digital signatures are typically randomized, and they depend on the message being signed. In particular, while copying over a handwritten signature from one document to another might make a convincing forgery if done skilfully, a similar approach cannot work with digital signature because a valid signature for one document is completely different from a valid signature for another document. This is a feature that makes digital signatures much harder to forge!

4.2.5 Zero-knowledge proofs

A *proof* is an object which convinces observers of the validity of a certain mathematical statement. Proofs can be interactive: a *proof protocol* is an interaction between two entities, a *prover* (Peggy) and a *verifier* (Victor), at the end of which the verifier outputs either "accept" or "reject" (depending on whether Peggy's actions convinced him of the validity of the specified mathematical statement).

Sometimes, it is desirable for Peggy that Victor learns whether or not a mathematical statement is true, but does not learn any other information. For example, suppose Peggy wants to prove to Victor that there are two prime numbers p, q such that $pq = 75359$. A simple way to prove it would be for Peggy to tell Victor the primes p and q: then, Victor can check for himself that p and q are both prime, and that $pq = 75359$. However, this method reveals to Victor extra information (namely, the actual values of p and q) beyond the truth of the statement

$$\exists \text{ primes } p, q \text{ such that } pq = 75359 \ .$$

A proof is said to be *zero-knowledge* if it reveals no information other than the validity of the statement that it is proving.

4.3 Cryptographic implementation

The scheme proposed below closely follows the STAR-Vote voting system [7], with some adaptations for the QV setting. We remark that our design is not specific to STAR-Vote, and there are several other voting schemes which use homomorphic encryption for vote tallying, which could be adapted similarly to support QV.

As in STAR-Vote, we assume the polling place has the following physical setup: there is a Check-In Station, a Judge Station, several Ballot Machines (each protected by a privacy-preserving screen), and a Ballot Box. Each Ballot Machine contains a secret signing key (following the approach of [10]).

We assume a homomorphic encryption scheme is used, and there is a publicly known public key K^* using which all votes are to be encrypted. The corresponding decryption key is shared, using a k-out-of-m threshold decryption scheme, between m *election trustees*.

Ballot Machine functionality In STAR-Vote, the Ballot Machines have an interface (such as a touchscreen) using which voters can input information, a ballot printer which prints human-readable paper ballots, and a receipt printer.[14] For the QV setting, we additionally require the Ballot Machines to have a slot that accepts cash (like an Automatic Teller Machine (ATM) which accepts cash deposits), and the ability to count and check authenticity of cash inputted via this slot. Each Ballot Machine has an attached lock-box for storing the cash, which can only be opened by the election trustees.

We require cash payments (and do not accept checks or debit cards) primarily for the purpose of anonymity: that is, so that payment amounts cannot be traced back to the individual who made the payment (even if the government issues a subpoena to the bank). Another reason to avoid checks and credit card payments is because these payments methods do not entail immediate transfer of funds: bouncing checks and/or defaults on credit card payments would be very tricky or impossible to deal with.

We now describe the steps that would be taken by a voter in our system.

Step 1. Identity check A voter (call her Alice) enters the polling place, goes to the Check-In Station, and identifies herself to a poll worker Bob. Bob checks that Alice is on the list of registered/eligible voters, and if so, Bob gives Alice an *anonymous token* which represents that she is authorized to vote.

Alice takes her token to the Judge Station and gives it to another poll worker Carol. Carol gives Alice a piece of paper with a 5-digit code on it.

Step 2. Create a vote Alice chooses a Ballot Machine at which to create her ballot. First, she inputs her 5-digit code to the Ballot Machine, which identifies her as a valid voter (but does not reveal her actual identity). Then, she inputs her vote to the Ballot Machine by indicating her preferred candidate and number of votes v. She is presented with a review screen so that she can check her selections before producing a paper record.

When she is ready, Alice instructs the Ballot Machine to print the paper record of her vote. The Ballot Machine then prints two items:

1. a paper ballot containing a human-readable description of Alice's vote and a random serial number, and

2. a take-home receipt that shows the voting terminal used, the time of the vote, and an encryption of her vote[15] using public key K^*.

[14] The printing of ballots and receipts could potentially be done by the same printing device.

[15] Note that in QV, there are two candidates to choose between. We represent a vote of

The paper ballot is printed face-up so that Alice can read it, and includes both values v and v^2. The receipt, however, must be face-down and inaccessible to Alice:[16] this could be achieved by printing it behind a glass barrier, for example.

Optional step. Challenge vote Once the Ballot Machine has produced the print-outs, Alice can choose to go ahead and cast her vote, or to *challenge* the printed vote. This follows the "cast-or-challenge" paradigm for voter-initiated audits, originally suggested by Benaloh [10]. Alice might choose the latter option because of an input error (or change of mind), or because she wants to check that the Ballot Machine indeed correctly printed her selected vote (rather than malfunctioning or perhaps maliciously recording the wrong vote). If she chooses to challenge the vote, the Ballot Machine appends to Alice's receipt a *proof* that the encryption was correctly formed. (Such a proof can only be produced if the encryption really was correctly formed; otherwise, the machine will be "caught cheating".) Then, the Ballot Machine outputs the two print-outs to Alice, and allows her to start the vote creation process again from scratch.

Step 3. Cast a vote If Alice decides to cast her vote, then the Ballot Machine prompts her to insert v^2 in cash into the payment slot. Before continuing, Alice should check that the requested amount of money is equal to the value v^2 which is printed on her paper ballot.[17]

Once she inserts the appropriate amount of cash,[18] the Ballot Machine produces a digital signature on the encrypted vote that has been printed on the receipt. The Ballot Machine appends the digital signature to the receipt, and outputs the printed ballot and receipt to Alice.[19] To finalize the casting of her vote, Alice must take her paper ballot and put it in the Ballot Box. The Ballot Box has a scanner that can read the serial number off the paper ballots, and communicates the serial numbers of cast paper ballots to the Judge Station so that the Judge Station can keep a record of which electronic (encrypted) ballots to include in the tally. An electronic ballot will not be included in the tally unless its corresponding paper ballot has been deposited in the Ballot Box.

---

magnitude m for candidate 1 by the number m, and we represent a vote of magnitude m for candidate 2 by the number $-m$.

[16] This requirement is to prevent some subtle coercion attacks which are discussed in [10], which we will not detail here.

[17] If not, she should abort and report the machine as malfunctioning. If Alice is able to compute squares, she should also check that the requested amount of money is indeed the square of the number of votes she decided to cast.

[18] Practically speaking, we would like the Ballot Machines to be kept well-stocked with change so that Alice can be given change if she doesn't have exactly the right amount. However, opening and closing the lock-box of the machine during the election is a security concern. Hopefully, it would be possible and sufficient to pre-load each machine with an ample stock of change before the election starts, clearly record the amount that was pre-loaded into each machine, and subtract these amounts from the total when the cash is eventually counted.

[19] Note that the receipts for spoilt ballots are not digitally signed by the Ballot Machine, so that Alice cannot claim falsely that a spoilt ballot was the real vote that she cast.

The encryption of each vote cast will be posted on a public Vote Bulletin Board (e.g. a website maintained by the election organizers), along with the voter's name.[20]

After placing her paper ballot in the Ballot Box, Alice goes home.

Step 4. Tally votes After all votes have been cast, it is time to tally them. Since the encryptions have been made in a homomorphic encryption scheme, we will use the homomorphic ciphertext addition method to compute a ciphertext c^* encrypting the sum $v^* = \sum_{i \in [n]} v_i$ of all the cast votes. Note that any observer can compute the ciphertext c^* for himself, based on the Vote Bulletin Board. The sign of the sum v^* (i.e. whether it is positive or negative) corresponds to the election outcome.

Finally, the election trustees collaborate to decrypt the ciphertext c^* to reveal the tally v^*. They publish v^*, along with a zero-knowledge proof that v^* is indeed the correct decryption of c^*.[21]

Step 5. Sum payments In addition, the sum of payments must be tallied. We make use of the homomorphic encryption scheme again, to compute a ciphertext $\hat{c}$ encrypting the sum $\hat{v} = \sum_{i \in [n]} v_i^2$ of the squares of all the cast votes. Again, note that any observer can compute $\hat{c}$ for himself, based on the Vote Bulletin Board. As above, the election trustees collaborate to decrypt the ciphertext $\hat{c}$ to reveal the election revenue $\hat{v}$. They publish $\hat{v}$, along with a zero-knowledge proof that $\hat{v}$ is indeed the correct decryption of $\hat{c}$.

Additionally, the lock-boxes of all the Ballot Machines are unlocked by the election trustees in a publicly observable way (e.g. on television), and the money in the lock-boxes is counted under public observation. The total amount of money in the lock-boxes should be at least $\hat{v}$.[22] If the amount of cash in the lock-boxes is less than $\hat{v}$, this is evidence that some money was mis-handled during the election!

Step 6. Audit For some time-period after the election outcome and revenue are announced, voters and/or third parties and/or election officials are encouraged to check the results for themselves to make sure that everything was properly done. (For example, by checking the values c^* and $\hat{c}$ were computed correctly as described above, checking the validity of the decryption proofs of

[20] This means that voters can, if they wish, check that their vote is recorded correctly on the bulletin board, by comparing to their paper receipt. Note that the public posting of these encryptions does not compromise ballot secrecy, since without the secret key, no information about the values of the votes is revealed.

[21] It is very important that the proof is zero-knowledge: in particular, seeing the proof does not enable observers to decrypt any ciphertexts other than c^*.

[22] Note that the amount of money may be more than $\hat{v}$, if some voters finalized and paid for their votes on the Ballot Machine, but then did not deposit their paper ballots in the Ballot Box, thus disqualifying their vote from being included in the tally. The phenomenon of voters leaving the polling place having only partially completed the voting process is known more generally as the "fleeing voter problem".

the election trustees, and/or (in the case of individual voters) checking that the votes were posted on the bulletin board match the votes on the paper receipts.)

If the audit fails, then the party which initiated the audit should publicly announce the failure. Then, everyone will all know that there was an error in the election procedure. The cause may be unknown: for example, it could be due to human error, software bugs, or a malicious attack. The election officials should have a prepared procedure for the case that the audit fails, which is appropriate to the context of the particular election; but this is beyond the scope of this paper.

Step 7. Reimbursements Each person who voted in the election (i.e. each person who is crossed off of Bob's list) is mailed a check for the amount of $\hat{v}/n$ where $\hat{v}$ is the election revenue (as above) and n is the total number of voters.

Verifiability The verifiability of this voting scheme follows from the following properties of the construction:

1. Each voter can have confidence that with high probability, his/her vote was correctly encrypted, since he/she has the option of challenging encrypted votes to make sure they were computed correctly.
2. Each voter can check that a copy of his/her vote encryption is correctly recorded on the bulletin board.
3. Any observer can check that the tallying of the votes which are on the bulletin board – and the tallying of the corresponding payment amounts – were done correctly, because
   (a) anyone can compute the ciphertext of the final tally for themselves, just from the bulletin board; and
   (b) the election trustees are required to provide a proof that they decrypted that ciphertext correctly. i
4. The storage and final counting of cash payments is publicly monitored, and can be checked (by any observer) against expected amount of cash based on the homomorphic tallying of squares of vote amounts.
5. From the total amount of election revenue, anyone can compute the reimbursement due to each voter, and each voter can check that his/her reimbursement check is for the correct amount.

From the above properties, it is reasonable to conclude that as long as sufficiently many[23] voters perform audits, the probability that an error in the election process is detected by an audit is overwhelming. Note that the dependence on the fraction of voters who perform audits is inherent to any audit-based voting scheme.

[23]Technically, for this implication to hold, the set of voters that audit needs to be not only sufficiently large, but also random (or at least, unpredictable to an adversary).

Robustness against false accusations This voting scheme's robustness against false accusations follows from:

1. The security of the digital signature scheme used by the Ballot Machines to sign the encrypted votes, which means that the authenticity of the votes listed on the bulletin board can be checked by any observer (by checking the validity of the signatures). In particular, if the Ballot Machines work correctly, no party can create incorrectly-encrypted votes and convincingly use them as evidence that the Ballot Machines did not work correctly, since this party will not be able to forge the Ballot Machine's digital signature.

2. Our assumption that reimbursement checks cannot be forged.

Secrecy The secrecy of this voting scheme follows from:

1. The security of the encryption scheme, which means that an observer of the bulletin board learns no information about the votes which were cast.[24]

2. The opacity and physical security of the lock-box, i.e.,

   (a) nobody can see inside it before it is opened; and

   (b) a voter who watches the poll worker physically place an amount of cash into the lock-box can be sure that that cash indeed entered the lock-box and will stay there until the lock-box is opened.

3. The anonymity of cash, i.e., the property that when the lock-box is opened, an observer cannot tell what are the amounts contributed by any individual voters.

A remark about poll-site ATMs While it would be very convenient for voters to have cash withdrawal machines available at the polling site (so that they can withdraw cash on the spot, and use it to vote), we strongly recommend against this as it would introduce a number of security vulnerabilities.

The record of withdrawal amounts from the cash withdrawal machines at the polling site would give a good indication of voters' payment amounts, which is a security concern (even when not linked to voters' identities), as discussed in Section 3.1. This makes the cash withdrawal machines a prime target for attacks/surveillance. Moreover, banks that see that their customers have withdrawn from a polling-site withdrawal machine could infer approximately how many votes that customer cast.[25]

[24] We believe that the standard security definition for homomorphic encryption would need further elaboration to accommodate the specific requirements for this system, which we do not detail here. However, we believe that the BGN cryptosystem [12] satisfies the enhanced requirements.

[25] Unless the customer deliberately takes precautions to prevent this, e.g. by withdrawing much more cash than they need for voting. Even if such precautions are taken, however, *some* information can be inferred about the payment amount (e.g. that it was probably less than or equal to the withdrawn amount). Moreover, we should not expect average users to take such precautions.

5 Implementation of electronic surveys

We now turn to the setting of *surveys*, where eligibility to participate is linked to a *virtual* identity (e.g. an email address). In this setting, we consider the problem of implementing a QV system digitally and entirely over a computer network such as the Internet. An example application could be market surveys.

5.1 Election/survey systems

Prior work has proposed auditable, anonymous election/survey systems which are very suitable for settings where there is a low risk of coercion and voter eligibility is linked to virtual identities.

Helios The Helios voting system [1] has been available online for the past few years, and has been used in practice in low-stakes elections (e.g. to elect members of professional societies). The Helios system, like STAR-Vote, is based on tallying encrypted votes using an additively homomorphic encryption scheme. For conducting electronic QV surveys, we suggest using Helios with the BGN encryption scheme described in Section 4.2.2. This allows for secrecy-preserving tallying not only of the encrypted votes $v_1, \ldots, v_n$ themselves, but also of the payment amounts $v_1^2, \ldots, v_n^2$.[26] In this section, we henceforth write "Helios" to mean "Helios instantiated with the BGN encryption scheme".

Importance of verifiability Recently, a system for conducting anonymous online surveys (called ANONIZE) has been proposed [28], which has the advantage of an easier (non-interactive) vote-casting procedure, at the expense of voter verifiability of the outcome. There are some settings in which the outcome of the survey need not be voter verifiable (or perhaps it is not even announced to voters), such as some market surveys, and it seems that ANONIZE was designed for such settings. However, in QV, the total amount of payments must always be verifiable by participants (otherwise, they cannot be certain that they have received their rightful refund), even if the outcome need not be verifiable. Thus, systems like ANONIZE are unsuitable for QV.

5.2 Payment systems

Traditional electronic payments Online election/survey systems which have been proposed in the literature do not integrate payments.[27] The main challenge of integrating QV payments into an election/survey system is in satisfying

[26] Section 4.3 described how to compute the tallies of votes and payments, in the context of STAR-Vote. These tallying methods are equally applicable in the context of Helios.

[27] Certain "paid survey" companies implement a system where survey takers are paid to take surveys. This is a completely different setting from the QV setting, since such companies do not generally provide anonymity for survey takers, and payments are made unilaterally from the survey organizer to the survey takers, and the payment amount is independent of the survey takers' choices in the survey.

the *secrecy* requirement that the amount of money that any individual voter pays into the election must be kept secret. Traditional financial institutions have to keep records of each transaction made by electronic methods (such as debit card payments and bank transfers), and the same is true of other online payment services such as Paypal and Venmo. This means that your bank (or Paypal/Venmo/etc.) knows how much you paid into the election, but also that that information is potentially available to the government (e.g. by subpoena). Hence, we do not consider payment via traditional financial institutions or online payment services to be a satisfactory solution.

Cryptocurrency The primary alternative means of electronically transferring money is cryptocurrency. Currently, Bitcoin [37] is by far the most widely used cryptocurrency in existence, with a market cap over $6 billion US dollars. Bitcoin is based on maintaining a *block chain* or public ledger recording every Bitcoin transaction that ever happened. An identity in Bitcoin is associated with a *public key*, and so the ledger contains transactions which are (essentially) of the form:

Public key K transferred N bitcoins to public key K'.

There are two main problems with using Bitcoin for QV payments. The first is that ownership of a public key is not anonymous: while there is no information built into the Bitcoin system which links public keys to real-world identities, recent research [35, 44] has shown that in practice, it is quite possible to deanonymize most Bitcoin public keys (and it is relatively easy for a government with subpoena power). The second problem is that *even if* individual ownership of public keys were perfectly anonymous, the public-ledger-based system has an inherent problem in that all payment amounts are publicly visible. Recall, from Section 3.1, the secrecy requirement that even the set of all vote magnitudes must be kept secret. The Bitcoin system seems simply incompatible with this sort of secrecy requirement.

Other cryptocurrency proposals have been made, which provide stronger anonymity/privacy guarantees than Bitcoin: notably, Zerocash [9] is an extension to Bitcoin[28] which allows for anonymous payments in which the payer and payee identities and the payment amounts are all provably hidden. At the time of writing, there is currently no fully-fledged deployment of Zerocash, but the currency is under active development by a recent startup (founded earlier this year, 2016) and available as an alpha release [53].

5.3 Integrating Helios and Zerocash for QV

In order to integrate Zerocash payments with Helios to build a secure electronic QV implementation, the election administrator needs a way of receiving payments. We assume that when a survey is set up, a new Zerocash identity is

[28] In fact, Zerocash can be implemented as an extension to any cryptocurrency with basic structural resemblance to Bitcoin. Many such currencies exist: they are often collectively referred to as "altcoins". The popularity of such altcoins is dwarfed by that of Bitcoin.

created to receive all the payments for that survey, and that sufficient information is published so that survey-takers can direct their payments to the correct recipient. A survey-taker must prove that he/she has made a payment of the appropriate amount, before being allowed to submit his/her survey responses. This can be done using zero-knowledge proofs, so that the survey-taker's Zerocash identity need not be revealed to the Helios system.[29]

We remark that when deploying such a system, additional precautions should be taken to ensure the anonymity of the information flowing *into* and *out of* the Zerocash network, since Zerocash's own anonymity guarantees only apply to interactions which take place within the Zerocash network. For this purpose, we suggest the use of technologies such as Tor [49] or I2P [29] which are designed to anonymize internet traffic.

Converting regular currency to election currency Zerocash supports all the secrecy and anonymity properties required by a secure QV implementation, so would be suitable for conducting electronic QV surveys, *if the currency were very widely used*. If the currency is insufficiently widely used, then many voters would be buying Zerocash with regular money solely for the purpose of the election. Then, simply observing the amounts of Zerocash that voters buy would be a relatively good indication of how many votes they cast.

Currently, even Bitcoin is not nearly widely enough used to mitigate such concerns. Thus, although Zerocash seems to be the electronic payment method which is most suitable for an electronic QV implementation, we stress that in practice, payment secrecy is likely to be compromised if the election currency is not in ubiquitous use across the voter population.

Not using currency at all? An alternative approach which is being tested experimentally by start-up Collective Decision Engines is not to use money at all, but to use an "election currency" which is only useful for survey responses. Each survey-taker is allocated the same number of election currency units to spend on the survey. Clearly, if there is only one choice to be made, then each survey-taker will be incentivized to spend all his votes on the candidate he prefers, and this will become equivalent to plurality voting.

However, Collective Decision Engines is designed for the setting of surveys which consist of multiple (two-candidate) questions. In this case, survey-takers will have to choose how to allocate their votes between the different questions, and will be incentivized to spend more votes on the issues they care more about. The result is still very different from using monetary payments, in that voters' choices can only reflect how much they care about certain questions on the survey relative to other questions on the survey. Collective Decision Engines claims that their findings indicate that market surveys conducted in this way

[29] From the perspective of practicality: the zero-knowledge proofs required here would be quite simple, so could be performed roughly as efficiently as the Zerocash payment itself (which, as mentioned earlier, is fast enough to be considered for practical deployment, e.g., by startup Zcash).

can still elicit useful information from survey-takers beyond that which would be elicited by traditional surveys.

5.4 Drawbacks of electronic implementation

We pause to note that existing security technologies are widely viewed as being inadequate at present to protect online elections from serious attack. See [23] for a discussion of the security issues arising in online elections: denial-of-service attacks (possibly for selected jurisdictions only), client-side malware, server-side malware, vote-selling, voter-coercion, and difficulty producing a evidence trail as to the correct outcome. For these reasons (and others), we do not support online voting schemes for actual governmental elections.

6 Designing robust refunds

While the concrete refund scheme suggested by Lalley and Weyl in [33] is to divide election revenue equally among all voters (Refund Rule 1, below), they remark that this particular refund rule is not essential ([33, footnote 6]):

> *"This particular refund rule is for concreteness and budget balance only, and plays no role in our results. A wide range of refund rules would work equally well."*

In fact, any *efficient and balanced* refund rule (see Definition 6.2) suffices for the proofs in [33] to go through.[30] We remark that the conditions that must be satisfied by a QV refund rule have not been formally specified in prior work.

Definition 6.1 (Basic refund rule)**.** *A* basic refund rule *is specified by a matrix* $\mathbf{R} = \{R_{i,j}\}_{i,j\in[n]} \in [0,1]^{n\times n}$ *where* $R_{i,j} \in [0,1]$ *is the fraction of funds paid in by voter* i *which is received by voter* j*.*

Refund Rule 1 (Equal division)**.** The *equal-division* refund rule is a basic refund rule $\mathbf{R}_{\text{eq}} = \{R_{i,j}\}_{i,j\in[n]}$ where for all $i, j \in [n]$, $R_{i,j} = 1/n$.

Definition 6.2 (Efficiency and balance)**.** *A basic refund rule* $\mathbf{R}$ *is* efficient and balanced *if it satisfies the following two properties:*

- Budget balance: $\forall i \in [n]$, $\sum_{j\in[n]} R_{i,j} = 1$.
- Efficiency: *There is a constant* $R^* \in [0,1)$ *such that* $\forall i \in [n]$, $R_{i,i} = R^*$.

Definition 6.1 gives a very simple definition of refund rule: in particular, the rule is linear and deterministic. Both of these conditions can be relaxed. In the scope of this work, we consider only linear refund rules.

[30] Private communication with Glen Weyl, 2016.

6.1 What determines the voter indices?

An alternative efficient and balanced basic refund rule is given below.

Refund Rule 2 (Cycle)**.** Voter 2 gets voter 1's payment, voter 3 gets voter 2's payment, and so on. That is, the *cycle* refund rule is a basic refund rule $\mathbf{R}_{\text{cyc}} = \{R_{i,j}\}_{i,j\in[n]}$ where for all $i \in [n]$,

$$R_{i,j} = \begin{cases} 1 & \text{if } j \equiv i+1 \pmod{n} \\ 0 & \text{otherwise} \end{cases}.$$

This immediately raises the question: *what determines the voter indices?* For example, with the cycle refund rule: if voters were assigned indices in the order that they check in at the polling place, then rational voters would try to position themselves behind wealthy-looking people in line, in the hope of getting a larger refund. Such behavior seems undesirable.

The problem above stems from the idea that voters may have (some degree of) control over their indices. Lalley and Weyl's original analysis [33] is in a model where voters cannot collude, does not (need to) address this issue, as the specific refund rule they use (i.e. equal division) is robust against such behavior because all voters' refund amounts are equal, independently of their indices.

6.1.1 Security against voter-index attacks

We propose a security definition for refund rules (Definition 6.3, below) which models the possibility that voters may be able to influence their indices (and the indices of others). Cryptographic security definitions generally characterize security against a "worst-case" adversary: this is considered good practice in order to maximally protect even against unanticipated types of attacks. Accordingly, Definition 6.3 assumes a powerful adversarial voter who can arbitrarily control the indices of all voters, and has detailed auxiliary information about their votes. We stress that we do not necessarily consider such a powerful adversarial voter to be realistic; we are just positing a worst-case scenario in order to strengthen the security definition.

Definition 6.3. *Let* $\mathbf{R}$ *be an efficient and balanced basic refund rule. Suppose that the amounts paid in by voters* $1, \ldots, n$ *are equal to* $p_1, \ldots, p_n$ *respectively. The refund rule* $\mathbf{R}$ *is said to be* secure against voter-index attacks *if a voter who knows* $p_1, \ldots, p_n$ *cannot get a higher reward even given the power to arbitrarily permute the indices of the voters.*

More formally: $\mathbf{R}$ *is* secure against voter-index attacks *if for all permutations* $\sigma \in \text{Sym}([n])$ *(where* $\text{Sym}([n])$ *denotes the symmetric group of* $[n]$*), for all* $j \in [n]$*, it holds that*

$$\sum_{i\in[n]} R_{\sigma(i),\sigma(j)} \cdot p_{\sigma(i)} = \sum_{i\in[n]} R_{i,j} \cdot p_i$$

It is straightforward to see that the equal-division refund rule already satisfies Definition 6.3. In fact, the equal-division rule is the *only* basic refund rule which is secure against voter-index attacks. However, we will now see that if we allow refund rules to be *randomized*, then there is a larger class of refund rules which are secure against voter-index attacks.

6.2 Randomized refund rules

In this subsection, we present a generalized definition of refund rules that allows for randomization, and a corresponding efficiency/balance condition and security definition.

Definition 6.4 (Randomized refund rule)**.** *A* randomized refund rule *is specified by a distribution $\mathcal{R} \in \Delta([0,1]^{n\times n})$ over basic refund rules, where $\Delta([0,1]^{n\times n})$ denotes the set of all distributions over $[0,1]^{n\times n}$. The randomized refund process is: sample a basic refund rule $\mathbf{R}$ from $\mathcal{R}$, then issue refunds according to $\mathbf{R}$.*

Definition 6.5 (Efficiency and balance)**.** *A randomized refund rule $\mathcal{R}$ is* efficient and balanced *if it satisfies the following two properties:*

- Budget balance: $\forall \mathbf{R} \in \mathrm{Supp}(\mathcal{R})$, $\forall i \in [n]$, $\sum_{j\in[n]} R_{i,j} = 1$.
- Expected efficiency:[31] *There is a constant $R^* \in [0,1)$ such that $\forall i \in [n]$, $\mathbb{E}[\mathbf{R}_{i,i}] = R^*$. The expectation is taken over the choice of $\mathbf{R}$ from $\mathcal{R}$.*

Note that when implementing a randomized refund rule, it is important to ensure that the sampling randomness is chosen honestly. One way to do this is to produce the randomness under public scrutiny, e.g. by rolling dice as the public watches (either in person or by television).

Moreover, the randomness must be sampled *after* all votes have been cast, as captured in the security definition that follows. This is a reasonable assumption even for a worst-case adversary, because a list of voter indices could be published (and signed by the election trustees for authenticity), before the refund amounts are computed.

Definition 6.6. *Let $\mathcal{R}$ be an efficient and balanced randomized refund rule. Suppose that the amounts paid in by voters $1,\ldots,n$ are equal to $p_1,\ldots,p_n$ respectively. The refund rule $\mathcal{R}$ is said to be* secure against voter-index attacks *if a voter who knows $p_1,\ldots,p_n$ cannot get a higher expected reward even given the power to arbitrarily permute the indices of the voters* before the randomization step of the refund rule.

More formally: $\mathcal{R}$ is secure against voter-index attacks *if for all permutations $\sigma \in \mathrm{Sym}([n])$, for all $j \in [n]$, it holds that*

$$\mathbb{E}\left[\sum_{i\in[n]} R_{\sigma(i),\sigma(j)} \cdot p_{\sigma(i)}\right] = \mathbb{E}\left[\sum_{i\in[n]} R_{i,j} \cdot p_i\right] ,$$

[31] For randomized refund rules, we believe requiring *expected* efficiency is sufficient since the QV analysis assumes voters to be risk-neutral, as also confirmed in private communication with Glen Weyl (2016).

where the expectation is taken over the randomness of the refund rule.

We highlight a couple of simple randomized refund rules which are efficient and balanced.

Refund Rule 3 (Lottery)**.** One person is chosen at random to receive the entire election revenue. That is, the *lottery* refund rule is a randomized refund rule $\mathcal{R}$ which is uniform over the following set of basic refund rules:

$$\left\{\mathbf{R} \in [0,1]^{n\times n} \ : \ \exists j^* \in [n] \text{ such that } \forall i \in [n],\ R_{i,j^*} = 1\right\} .$$

Refund Rule 4 (Random swaps)**.** Voters are randomly paired up, and each voter's payment is received (in full) by his/her "partner". That is, the *random-swaps* refund rule is a randomized refund rule $\mathcal{R}$ which is uniform over the following set of basic refund rules:[32]

$$\left\{\mathbf{R} \in [0,1]^{n\times n} \ : \ \exists \sigma \in \mathrm{Sym}_2([n]) \text{ s.t. } \forall i \in [n], R_{i,j} = \begin{cases} 1 & \text{if } \sigma(i) = j \\ 0 & \text{otherwise} \end{cases}\right\} ,$$

where $\mathrm{Sym}_2([n]) \subset \mathrm{Sym}([n])$ denotes the set of permutations of $[n]$ that decompose into a product of disjoint transpositions (i.e., 2-cycles). In other words, $\mathrm{Sym}_2([n])$ is the set of permutations which consist solely of swaps between disjoint pairs of elements.

In fact, *any* basic refund rule $\mathbf{R}$ which is efficient and balanced (but not necessarily secure against voter-index attacks) can be transformed into a *randomized* refund rule $\mathcal{R}$ which is efficient and balanced and also secure against voter-index attacks, by the following simple transformation: let $\mathcal{R}$ be the uniform distribution over

$$\left\{\mathbf{R}' \in [0,1]^{n\times n} \ : \ \exists \sigma \in \mathrm{Sym}([n]) \text{ such that } \forall i,j \in [n],\ R'_{i,j} = R_{\sigma(i),\sigma(j)}\right\} .$$

Finally, we remark that security against voter-index attacks is not the only relevant consideration in designing a robust refund rule. For example, it is also important that the refund rule preserves *secrecy*. Note that in the case of the lottery rule, this does not pose a concern; but in the case of the random swaps, we would like the voters not to know the identity of their "swap-partner" since otherwise they would learn the amount of their partner's vote.

Why consider randomized refund rules? One advantage of randomized refund rules is that they may be much simpler to administer: for example, the lottery refund rule only requires reimbursement of a single voter, rather than managing the distribution of checks to every voter who turned out to vote. Methods for securely running lotteries between a distributed network of participants are known based on a cryptographic tool called *secure multi-party*

[32] As written, the rule only works when the number of voters is even. We wrote it thus for simplicity, and note that it can straightforwardly be extended to cover the odd case too.

computation [26, 8, 19], and recent work also shows specific practical methods for running lotteries over the Bitcoin network [3].

Another potential advantage is that they may be able to be realized with weaker trust assumptions, and therefore be administered in more adversarial environments. For example, the random-swaps rule may be useful in a setting where the cash flow cannot be reliably monitored on television from beginning to end. This might mean that voters do not trust the election authority to handle their cash correctly, but the election may still be able to go ahead since the exchange of cash can occur solely between pairs of voters, and never go through the central authority at all.[33]

6.3 Collusion and refund rules

Another potentially problematic issue when designing robust refund rules is collusion between voters. We exemplify this by way of another simple refund rule, given below.

Refund Rule 5 (Neighbor)**.** Voters 1 and 2 swap payments, voters 3 and 4 swap payments, and so on.[34] That is, the *neighbor* refund rule is a basic refund rule $\mathbf{R}_{\text{nbr}} = \{R_{i,j}\}_{i,j\in[n]}$ where for all $i \in [n]$,

$$R_{i,j} = \begin{cases} 1 & \text{if } j \equiv i+1 \pmod N \text{ and } i \text{ is odd} \\ 1 & \text{if } j \equiv i-1 \pmod N \text{ and } i \text{ is even} \\ 0 & \text{otherwise} \end{cases}.$$

With the neighbor refund rule, two colluding friends with the same political inclination could stand next to each other in line and agree on an arbitrarily large number of votes to cast, safe in the knowledge that their net monetary loss will be 0 since they will just swap payments in the end.

This type of attack is in a materially different category from the attacks on refund rules which were described earlier in this section, because the adversarial voter aims to manipulate the *election outcome* to his/her advantage, rather than aiming to manipulate his/her *refund amount* as in the previously described attacks. Nonetheless, the specific attack we described against the neighbor refund rule can be prevented by the randomization techniques from Section 6.2.

Collusion between voters is an issue that is not addressed by Lalley and Weyl's original analysis [33] and is an inherent issue completely independent of voter-index manipulation. For example, because QV requires quadratic pricing

[33] Designing such a voting scheme in detail is tricky, and we do not currently have a detailed design. Some challenges include: (a) even if exchange of cash occurs solely between pairs of voters, it must somehow be ensured that each voter cannot cast more votes than he/she paid for, and (b) the pairings must be chosen randomly, so voters need a way of reliably transferring cash to a specific random stranger. Still, these issues seem like they could be resolved, at least for small-scale local elections where all voters are geographically close and the election can occur within a short time-frame.

[34] Technically, the refund rule as stated only works when n is even. The rule can straightforwardly be modified to support odd n too.

of votes *per individual*, two colluding individuals can buy v votes for a lower price than one individual alone.

Issues of collusion to manipulate the election outcome are beyond this scope of this paper. For a more detailed discussion of this topic, we refer to a recent paper by Weyl [52] which partially addresses the issue of collusion in QV.

7 Conclusion

A secure implementation of quadratic voting requires the implementation of two of the most challenging functionalities even in the stand-alone setting: voting, and payments.

This paper provides an overview of the challenges that must be addressed in a secure implementation of quadratic voting, and suggests how certain physical and cryptographic technologies might help to secure such an implementation.

Our presentation considers as separate cases in-person pollsite voting and remote voting. Our first pollsite voting proposal uses a quantity of wax sold to the voter to represent a quantity of votes; wax has the nice property that chunks of it can be melted together to provide anonymity about the chunk sizes. Our second pollsite voting scheme uses cryptographic techniques, including (additive, single-multiplication) homomorphic encryption, to achieve similar properties. Both pollsite voting proposals use cash for payments. Our remote voting proposal uses homomorphic encryption in conjunction with payments made through a cryptocurrency with strong anonymity properties.

It must be emphasized, however, that remote voting over the internet introduces a myriad of security issues, such as denial-of-service attacks, and there is as yet no way to fully secure internet voting, for any voting system. See the Overseas Vote Foundation report [23] for a more detailed study of the problems intrinsic to trying to vote over the Internet.

In addition, we are unable to provide any recourse against the problem of voter collusion (pooling their resources to buy votes more cheaply); this seems like an intrinsic issue for QV.

But in low-stakes scenarios, where coercion and vote-buying seem unlikely, the approaches we suggest may be applicable, just as, for example, the Helios system [1] seems usable when such attacks seem highly improbable.

Finally, we suggest some extensions to the basic quadratic voting framework. In particular, we study and characterize robust refund rules, and introduce some novel randomized refund rules (like the "lottery" method) that satisfy our refined conditions for acceptable QV refund rules, and which may well be worth further investigation due to their simpler administrative requirements and entertainment value for voters.

Acknowledgments

This work was supported by the Center for Science of Information STC (CSoI), an NSF Science and Technology Center, under grant agreement CCF-0939370. We also received support from the MACS project NSF grant CNS-1413920 and a Simons Investigator Award Agreement dated 2012-06-05.

We would also like to thank Glen Weyl for encouraging us to work on these issues, and for his helpful feedback at many points during this research.

References

[1] Ben Adida. Helios: Web-based open-audit voting. In van Oorschot [51], pages 335–348.

[2] Alfred V. Aho, editor. *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*. ACM, 1987.

[3] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. Secure multiparty computations on Bitcoin. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, SP '14, pages 443–458, Washington, DC, USA, 2014. IEEE Computer Society.

[4] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In Sadeghi et al. [46], pages 1087–1098.

[5] O. Baudron, P.-A. Bouque, D. Pointcheval, G. Poupard, and J. Stern. Practical multi-candidate election system. In N. Shavit, editor, *Proc. 20th ACM PODC '01*, pages 274–283. ACM, 2001.

[6] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and noninteractive anonymous credentials. In Canetti [17], pages 356–374.

[7] Susan Bell, Josh Benaloh, Michael D. Byrne, Bryce Eakin, Philip Kortum, Neal McBurnett, Olivier Pereira, Philip B. Stark, Dan S. Wallach, Gail Fisher, Julian Montoya, Michelle Parker, and Michael Winn. STAR-Vote: A secure, transparent, auditable, and reliable voting system. In *Presented as part of the 2013 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections*, Berkeley, CA, 2013. USENIX.

[8] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 1–10. ACM, 1988.

[9] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from Bitcoin. In *2014 IEEE Symposium on Security and*

Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014, pages 459–474, 2014.

[10] Josh Benaloh. Ballot casting assurance via voter-initiated poll station auditing. In Martinez and Wagner [34].

[11] Josh Benaloh, Ronald L. Rivest, Peter Y. A. Ryan, Philip B. Stark, Vanessa Teague, and Poorvi L. Vora. End-to-end verifiability. *CoRR*, abs/1504.03778, 2015.

[12] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Kilian [32], pages 325–341.

[13] Stefan Brands, Liesje Demuynck, and Bart De Decker. A practical system for globally revoking the unlinkable pseudonyms of unknown users. In Pieprzyk et al. [41], pages 400–415.

[14] Stefan Brands and Frédéric Légaré. Digital identity management based on digital credentials. In Schubert et al. [47], pages 120–126.

[15] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Pfitzmann [40], pages 93–118.

[16] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Franklin [24], pages 56–72.

[17] Ran Canetti, editor. *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, volume 4948 of *Lecture Notes in Computer Science*. Springer, 2008.

[18] Richard Carback, David Chaum, Jeremy Clark, John Conway, Aleksander Essex, Paul S. Herrnson, Travis Mayberry, Stefan Popoveniuc, Ronald L. Rivest, Emily Shen, Alan T. Sherman, and Poorvi L. Vora. Scantegrity II municipal election at Takoma Park: The first E2E binding governmental election with ballot privacy. In Ian Goldberg, editor, *Proceedings USENIX Security 2010*. USENIX, 2010.

[19] David Chaum, Claude Crépeau, and Ivan Damgard. Multiparty unconditionally secure protocols. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 11–19, New York, NY, USA, 1988. ACM.

[20] Ivan Damgård, Mads Jurik, and Jesper Buus Nielsen. A generalization of paillier's public-key system with applications to electronic voting. *Int. J. Inf. Sec.*, 9(6):371–385, 2010.

[21] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, Jul 1985.

[22] Caroline Fontaine and Fabien Galand. A survey of homomorphic encryption for nonspecialists. *EURASIP J. Inf. Secur.*, 2007:15:1–15:15, January 2007.

[23] Overseas Vote Foundation. The future of voting: End-to-end verifiable internet voting specification and feasibility assessment study. `https://www.usvotefoundation.org/news/E2E-VIV-press`, 2015.

[24] Matthew K. Franklin, editor. *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*. Springer, 2004.

[25] Craig Gentry. Computing on the edge of chaos: Structure and randomness in encrypted computation. *IACR Cryptology ePrint Archive*, 2014:610, 2014.

[26] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In Aho [2], pages 218–229.

[27] Gurchetan S. Grewal, Mark D. Ryan, Liqun Chen, and Michael R. Clarkson. Du-Vote: Remote electronic voting with untrusted computers. In *Prof. 28th IEEE Computer Security Foundations Symposium*. IEEE, 2015.

[28] Susan Hohenberger, Steven Myers, Rafael Pass, and Abhi Shelat. An overview of ANONIZE: A large-scale anonymous survey system. *IEEE Security & Privacy*, 13(2):22–29, 2015.

[29] I2P anonymous network. `https://geti2p.net/`.

[30] Douglas W. Jones and Barbara Simons. *Broken Ballots — Will Your Vote Count?* CSLI, 2012.

[31] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2007.

[32] Joe Kilian, editor. *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, volume 3378 of *Lecture Notes in Computer Science*. Springer, 2005.

[33] Steven P. Lalley and E. Glen Weyl. Quadratic voting. 2016. Available online at SSRN: `http://ssrn.com/abstract=2003531`.

[34] Ray Martinez and David Wagner, editors. *2007 USENIX/ACCURATE Electronic Voting Technology Workshop, EVT'07, Boston, MA, USA, August 6, 2007*. USENIX Association, 2007.

[35] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. A fistful of bitcoins: characterizing payments among men with no names. In Papagiannaki et al. [39], pages 127–140.

[36] C. Moore, M. O'Neill, E. O'Sullivan, Y. Dorz, and B. Sunar. Practical homomorphic encryption: A survey. In *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*, pages 2792–2795, June 2014.

[37] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009. `http://bitcoin.org/bitcoin.pdf`.

[38] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'99, pages 223–238, Berlin, Heidelberg, 1999. Springer-Verlag.

[39] Konstantina Papagiannaki, P. Krishna Gummadi, and Craig Partridge, editors. *Proceedings of the 2013 Internet Measurement Conference, IMC 2013, Barcelona, Spain, October 23-25, 2013*. ACM, 2013.

[40] Birgit Pfitzmann, editor. *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, volume 2045 of *Lecture Notes in Computer Science*. Springer, 2001.

[41] Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors. *Information Security and Privacy, 12th Australasian Conference, ACISP 2007, Townsville, Australia, July 2-4, 2007, Proceedings*, volume 4586 of *Lecture Notes in Computer Science*. Springer, 2007.

[42] Stefan Popoveniuc and Jonathan Stanton. Undervote and pattern voting: Vulnerability and a mitigation technique. In Martinez and Wagner [34].

[43] Range voting. `http://rangevoting.org`.

[44] Dorit Ron and Adi Shamir. Quantitative analysis of the full Bitcoin transaction graph. In Sadeghi [45], pages 6–24.

[45] Ahmad-Reza Sadeghi, editor. *Financial Cryptography and Data Security - 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers*, volume 7859 of *Lecture Notes in Computer Science*. Springer, 2013.

[46] Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors. *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*. ACM, 2013.

[47] Sigrid E. Schubert, Bernd Reusch, and Norbert Jesse, editors. *Informatik bewegt: Informatik 2002 - 32. Jahrestagung der Gesellschaft für Informatik e.v. (GI), 30. September - 3.Oktober 2002 in Dortmund*, volume 19 of *LNI*. GI, 2002.

[48] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[49] Tor project: Anonymity online. `https://www.torproject.org`.

[50] V. Vaikuntanathan. Computing blindfolded: New developments in fully homomorphic encryption. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 5–16, Oct 2011.

[51] Paul C. van Oorschot, editor. *Proceedings of the 17th USENIX Security Symposium, July 28-August 1, 2008, San Jose, CA, USA*. USENIX Association, 2008.

[52] E. Glen Weyl. The robustness of quadratic voting. 2015. Available online at SSRN: `http://ssrn.com/abstract=2571012`.

[53] Zcash. `https://z.cash`.