End-to-End Verifiable Quadratic Voting with Everlasting Privacy

Olivier Pereira[1] and Peter B. Rønne[2]

[1] Université catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium, olivier.pereira@uclouvain.be
[2] SnT, University of Luxembourg, Luxembourg, peter.roenne@uni.lu

Abstract. Quadratic voting is an intriguing new method for public choice suggested by Lalley and Weyl, which they showed to be utilitarian efficient. Voters are given a budget of credits and can assign each of the candidates a (perhaps negative) value, where the price paid for their voting choice is the sum of the squared values. From a security viewpoint, we generally request elections to be private and have integrity, and even further (end-to-end) verifiability which entails public bulletin boards. Such public data might be troublesome when considering future adversaries capable of breaking current cryptographic primitives, either due to computational power advances, broken primitives or scientific breakthroughs. This calls for election schemes with everlasting privacy and perfectly private audit trails. In the case of quadratic voting this is even more crucial since budget balances have to be linked between elections in a verifiable way, and revealing old budget values partially break privacy in later elections. In this paper, we suggest an efficient construction of electronic quadratic voting with end-to-end verifiability and a perfectly private audit trail inspired by the methods of Cuvelier, Pereira and Peters, but adapted to include the quadratic relations and keeping budget balances everlasting private.

1 Introduction

Finding good public choice methods is a notoriously hard problem. Recently a novel intriguing approach has appeared: quadratic voting [16]. The quadratic voting method works by providing the voter with a budget b of credits for buying votes, however, the voting credit does not have to be connected to a real financial currency. The peculiarity is that a voter casting v votes for a candidate has to pay a quadratic amount v^2 of credits for this choice. That is, the voter assigns vote values $v_1, \ldots, v_c$ to the c candidates, or choices, and has to pay the sum of squared values which have to be within budget

$$v_1^2 + \cdots + v_c^2 \leq b \ .$$

The advantage of quadratic voting is that it theoretically satisfies utilitarian efficiency at least in the asymptotic case [16], and in the finite case the inefficiency is suppressed by the number of voters, see also [8]. To put it differently,

the quadratic pricing gives incentive for the voter to buy a number of votes corresponding to her internal value.

An enlightening example demonstrating the effect of quadratic voting on real users can be found in [19]. Here voting was not directly considered but rather closely related surveys. In a combined between-groups and within-subjects study, participants were asked about their opinion on 10 proposals. One group of participants gave answers on a Likert 7-choice scale ranging from "Very strongly against" over "Neutral" to "Very strongly in favor" whereas another group gave responses using quadratic voting with a total budget of 100 credits for all answers i.e. being able to vote in the range $\{-10, \ldots, 0, \ldots, 10\}$. As is intuitive the quadratic voting resulted in much less extreme answer but, further, the answers were also much closer to being normal distributed. The quasi-normal distribution could be an indication that the answers were closer to expressing the true internal value.

Once the vote is complete, the collected payments are redistributed among the voters. The method suggested by Lalley and Weyl [16] is to split the revenue of the election evenly among the voters. However, other solutions have been proposed, including lotteries [17]. The actual choice is not essential for the utilitarian efficiency. For simplicity we will focus on the even split of revenue in this paper.

Quadratic voting is also an interesting challenge from a security viewpoint, as we have to cryptographically deal with squared values and checks of budget balances. A first solution for running end-to-end verifiable elections with quadratic voting is described by Park and Rivest [17]. Here security properties of voting schemes are discussed, and the importance of budget privacy is stressed, especially if revealing individual votes due to Italian attacks. Our scheme uses homomorphic tallying, partially sidestepping the Italian attacks. Still, budget balances should be kept private by the voters, as they could result in vote privacy leaks. Note that a very small leak in privacy is unavoidable since we reveal the total revenue by refunding it to the voters. Park and Rivest also analyze strategic voting and refunding rules for quadratic voting, and further suggest schemes for in-person and electronic voting, with cryptography based on the BGN encryption scheme [6], which allows to calculate squares of encrypted values, and to further homomorphically add those squares. Regarding the handling of the payments, it is also mentioned as a possible option to use an anonymous cryptocurrency such as Zerocash [20].

We create a different solution, which offers several additional benefits:

- Our protocol offers a perfectly private audit trail (PPAT), that is, all the data needed for public verifiability perfectly hide all the votes and budgets. The previous solution requires publishing ciphertexts that would eventually leak vote content.
- Our protocol is compatible with traditional threshold key generation protocols in the discrete log setting [18, 15], even in the malicious setting. The BGN scheme requires the use of an RSA modulus with unknown factorisa-

tion, which is considerably more challenging to obtain (see discussion in [6] for instance).
– Our solution is quite efficient: voter computation takes place in prime order groups on elliptic curves (e.g., BN curves [2]), and only requires to compute one pairing per election. The BGN based solution requires to compute on curves with a modulus that has the size of an RSA modulus, and requires the evaluation of pairings for each vote.

We believe that everlasting privacy of the audit data is an important improvement for any secure election scheme: we want these data to be widely available to the public, but then need to take care that votes do not leak in the future. An adversary may benefit from the ever-increasing efficiency improvements in computing, the breaking of believed-secure cryptographic assumptions or technical breakthroughs such as quantum computers. However, in quadratic voting such future-proofing is even more essential, since the budget can be carried over from election to election. Thus breaking the privacy of earlier elections might (partially) leak the later budget in the present of the future adversary.

We also conjecture that handling the payments inside the voting system (compared to relying on an externally managed cryptocurrency) is an interesting feature: this avoids mixing the systems of incentive that come with cryptocurrencies with those in the election process, and it also makes it easier to control that all voters start from an equal budget.

The outline of the paper is as follows. In Section 2 we present the necessary cryptographic tools, that will be used in the cryptographic protocol presented in Section 3. In Section 4 we discuss the security properties of the protocol. We end with a conclusion and discussion of future research directions.

2 Background and Cryptographic Tools

2.1 Commitment Consistent Encryption

The first component of our quadratic voting protocol is commitment consistent encryption (CCE) [12], a cryptographic tool that was proposed to facilitate the design of universally verifiable voting schemes with a perfectly private audit trail.

A CCE scheme is a traditional public key encryption scheme offering an extra feature: from any ciphertext, it is possible to derive a perfectly hiding commitment, as well as an opening of that commitment to the value that is encrypted. The commitment derivation operation, DCom, only requires using the public key, while the computation of the opening, Open requires the secret key.

It is convenient to have, associated to a CCE scheme, the possibility to use efficient proof systems, which can serve several purposes, and which we will need in our application:

– A proof of validity of a ciphertext, that guarantees election organisers that the Open operation would succeed without performing it. Quite often, ciphertexts are never decrypted but rather homomorphically combined, e.g.,

in order to add votes. A single invalid ciphertext would then suffice to make it impossible to open the election result.
- A proof of knowledge of the plaintext. This can be used to make ciphertexts non-malleable [22, 5], and avoid attacks on privacy.
- A proof of validity of a plaintext, which guarantees that the plaintext that is encrypted encodes a valid vote.

These will be discussed in the next subsection.

We use the PPATS encryption scheme of Cuvelier et al. [12], which is described in Figure 1. This scheme works in an asymmetric bilinear group setting, which can be obtained using BN curves [2] for instance. In the group $\mathbb{G}_1$, two random generators (g_1, h_1) are given in order to produce Pedersen-like commitments and, in the group $\mathbb{G}_2$, an ElGamal key (g_2, h_2) is produced. The main twist of the scheme is to open a commitment $g_1^v h_1^r$ using g_2^r and to verify it using the pairing operator, instead of using r directly as with Pedersen commitments: the pairing makes it possible to verify, after removal of the g_1^v term, whether $e(h_1^r, g_2) = e(h_1, g_2^r)$. Using this alternate way of opening the commitment offers two benefits: (i) the opening is now a group element, which can be conveniently encrypted using the ElGamal key (ii) all the secret values are in exponents, which eases compatibility with traditional, efficient, Σ-protocols.

PPATS encryption

$\mathsf{Setup}(1^\lambda)$ Return, as a public parameter pp, type-3 pairing-friendly groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of prime order q s.t. $|q| = \lambda$, together with random generators g_1, h_1 of $\mathbb{G}_1$ and g_2 of $\mathbb{G}_2$. We assume that pp is available to all other algorithms.

$\mathsf{Gen}(1^n)$ Generate an ElGamal public encryption key $\mathsf{pk} = h_2 = g_2^x$. The secret key is the uniformly random $\mathsf{sk} = x \in \mathbb{Z}_q$.

$\mathsf{Enc}_{\mathsf{pk}}(v)$ Encrypt vote v as $(d, c_1, c_2) = (g_1^v h_1^r, g_2^s, g_2^r h_2^s)$, using uniformly random $(r, s) \leftarrow \mathbb{Z}_q^2$.

$\mathsf{Dec}_{\mathsf{sk}}(d, c_1, c_2)$ Return the discrete logarithm of $e(h_1, c_1^x / c_2) e(d, g_2)$ in basis $e(g_1, g_2)$.

$\mathsf{DCom}_{\mathsf{pp}}(d, c_1, c_2)$ Derive and return the perfectly hiding commitment d.

$\mathsf{Open}_{\mathsf{sk}}(d, c_1, c_2)$ The commitment opening is computed as $a = c_2 / c_1^x$.

$\mathsf{Vrfy}_{\mathsf{pk}}(d, v, a)$ (v, a) is an opening of d iff $e(h_1, a) = e(d / g_1^v, g_2)$.

Fig. 1. The PPATS encryption scheme

2.2 Sigma protocols

A Sigma protocol [13], or Σ-protocol, for a relation R enables a prover P to convince a verifier V that he knows a witness w for a statement x such that $(w, x) \in R$.

Sigma-protocols are structured as follows: based on a joint input x, P sends a commitment a to V, who answers with a uniformly random challenge e and,

finally, P sends a response f. Based on this response, V accepts or rejects the proof.

Σ-protocols exhibit the following properties:

Completeness If P and V follow the protocol honestly and if P actually knows a witness w for the statement x, then V accepts the proof.

Special honest verifier zero-knowledge There is a simulator S that, from any valid statement x and challenge e from the set of possible challenges, is able to produce a full valid protocol transcript (a, e, f). If e is uniformly distributed, then this transcript is distributed exactly like a real protocol execution. (Note that no valid witness w for x is given to S.)

Special soundness From any two valid proof transcripts (a, e_1, f_1) and (a, e_2, f_2) for a statement x, with a single commitment a and two distinct challenges $e_1 \neq e_2$, it is possible to extract a witness w s.t. $(w, x) \in R$.

Σ-protocols come with two interesting features: (i) They can be efficiently turned into non-interactive zero-knowledge in the random oracle model thanks to the Fiat-Shamir heuristic [14, 5]; (ii) their perfect ZK property makes them suitable to be published as part of a perfectly private audit trail.

We need to use several standard Σ-protocols, which we list below.

Opening of a commitment. We use a protocol $\pi^b_{op}(c)$ to prove knowledge of an opening of a commitment c to a value b s.t. $c = g^b h^r$. This can be achieved using Schnorr's protocol [21], which takes a single exponentiation.

CCE ciphertext validity. We use a protocol $\pi_{va}(c)$ that proves the validity of a PPATS ciphertext c, by demonstrating the knowledge of the vote v and randomness (r, s) used to produce c. This protocol guarantees to the talliers that they will be able to run the tallying protocol successfully, as explained above. Such a protocol has been constructed for PPATS by Cuvelier et al. [12] and requires 2 exponentiations in $\mathbb{G}_1$ and 3 exponentiations in $\mathbb{G}_2$.

Range proof. We use a protocol $\pi^n_r(c)$ that proves the ability to open a commitment $c = g^v h^r$ on a vote v that is included in the range $[0, 2^n]$, with $n < \log q - 1$. Note that the notion of "positive" has a slightly unusual meaning here, as the values we are committing to lie in $\mathbb{Z}_q$; this is the reason of our upper bound on n, which guarantees that values do not "overflow" to negative values, interpreted as those above $(q-1)/2$.

This protocol will be used by the voters to prove that they are not overspending, that is, that their budget after each vote remains positive.

Many such proofs have been proposed, with their efficiency differing depending on the value of the range upper bound 2^n (among other factors). As we work in prime order groups and our range upper bound is a power of 2, we simply rely on the protocol by Bellare and Goldwasser [3]. This protocol makes a sequence of n commitments $c_0, \ldots, c_n$ on the individual bits $v_0, \ldots, v_i$ of the binary decomposition of v, proves that each commitments actually commit to bits, and then show that $c / \prod c_i^{2^i} = h^s$ for a known s.

If the 0/1 proofs are made using the disjunctive proofs of Cramer et al. [10], which takes 3 exponentiations in the group in which the commitment lies, then the total cost of such a proof is (i) n exponentiations for the bit commitments, (ii) $3n$ exponentiations for the proofs that they can be opened on bits, (iii) 1 exponentiation for the final proof on s. The total is then $4n+1$ exponentiations.

2.3 Proof of square

Finally, a key ingredient of our quadratic voting protocol is a proof π_{sq} that one can open two perfectly hiding commitments on values such that one is the square of the other. This will be used by the voters to show that they commit on an accurate payment based on their vote.

Such a proof can be achieved using the usual technique systematically described by Camenisch [7]. We propose here a slightly more efficient method, which we detail.

Suppose that P publishes two commitments c_1 and c_2 and wishes to demonstrate that he knows pairs (v_1, r_1) and (v_2, r_2) such that $c_1 = g^{v_1}h^{r_1}$, $c_2 = g^{v_2}h^{r_2}$ and $v_2 = v_1^2 \bmod q$. He can then follow the Square protocol depicted in Figure 2.

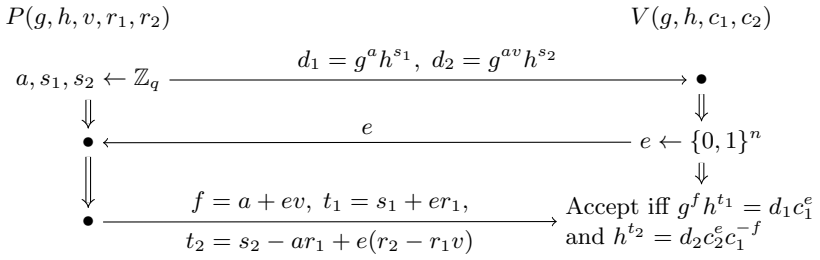

Fig. 2. Square: a Σ-protocol for commitments on values in quadratic relation.

Theorem 1. *The protocol $\pi_{sq}(c_1, c_2)$ described in Figure 2 is a Σ-protocol for the relation $\{v, r_1, r_2 | c_1 = g^v h^{r_1}, c_2 = g^{v^2} h^{r_2}\}$.*

Proof. We show the completeness, the special soundness and the perfect honest verifier ZK of the protocol.

Completeness: The perfect completeness of the protocol follows from the inspection of the verification equations. In particular, we see that $d_2 c_2^e c_1^{-f}$ is a commitment on $av + ev^2 - v(a + ev) = 0$.

Special soundness: Let us imagine that we have two valid transcripts for the same c_1, c_2, d_1, d_2, that is, we have (e, f, t_1, t_2) and (e', f', t'_1, t'_2) that are both consistent with the verification equations. Dividing the two versions of the verification equations gives: $g^{f-f'} h^{t_1 - t'_1} = c_1^{e-e'}$ and $h^{t_2 - t'_2} = c_2^{e-e'} c_1^{f'-f}$.

The first of these equations shows that $v = \frac{f-f'}{e-e'}$ and $r_1 = \frac{t_1-t'_1}{e-e'}$ are a valid opening of c_1. Inserting the extracted v in the second equation gives $c_2^{e-e'} = c_1^{v(e-e')} h^{t_2-t'_2}$. Isolating c_2, we can open it on the pair $(v^2, r_1 v + \frac{t_2-t'_2}{e-e'})$, and observe that the second element of that pair equals r_2.

Special HVZK: Given any e, we can select f, t_1, t_2 uniformly at random in $\mathbb{Z}_q$, then compute d_1 and d_2 from the verification equations. If e is uniformly random, then it is distributed as in the real protocol. The uniform selection of a, s_1, s_2 in a real execution guarantees that f, t_1, t_2 are random in the absence of d_1 and d_2, and those two commitments only enforce the verification equations. Hence, the simulated view is distributed exactly as a real one. □

The cost of this protocol is 3 exponentiations (considering that v is small), which is slightly better than the 4 exponentiations that would be obtained using the more common approach [7]. This may make this protocol of independent interest.

3 Verifiable Quadratic Voting

We now describe the steps of our quadratic voting protocol in detail. The main participants are the Election Authority with a set of Tally Tellers jointly holding the election secret key, the Voters and a Public Bulletin Board used to publish and verify the outcome of the election.

Our election setting and adversarial model is standard (we will discuss security in the next section) and similar to the one used by Cramer et al. [11] or Helios 2.0 [1].

The Election Authority orchestrates the election, publishing the questions and election public parameters (keys) on the public bulletin board, which is assumed to behave as a trustworthy broadcast channel. The Election Authority also handles the voter lists, and offers authenctication services to the voters if needed.

We aim for an end-to-end verifiable protocol: the election result should be verifiable without requiring to trust any particuliar entity or entities.

Regarding privacy, we want that votes remain computationally secret in front of the Tally Tellers: Tally Tellers would only be able to break privacy if a computational assumption is broken, or if enough of them are malicious (the threshold can be arbitrarily chosen). Furthermore, we want that all the data published on the Bulletin Board guarantee the perfect privacy of the votes: someone who can only access the Bulletin Board should never be able to learn the votes, independently of the falsification of any computational assumption.

3.1 Election Setup

Parameter generation. Given a security parameter n, generate public parameters $\mathsf{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, q, g_1, h_1, g_2)$ and a key pair $(\mathsf{pk}, \mathsf{sk}) = (h_2, x)$ for the PPATS encryption scheme. The key pair may be generated in a distributed or threshold

fashion by the Tally Tellers. We define the commitment key as $\mathsf{cpk} = (g_1, h_1)$, taken from pp.

Initialization. Before the voting starts, a public bulletin board is initialized, the public parameters and keys $(\mathsf{pp}, \mathsf{pk}, \mathsf{cpk})$ are published there, together with the inital budget of every voter b and the first election question.

We assume that b is a reasonably small value, e.g., $b < 2^{20}$, so that it will be possible to run the PPATS decryption of sum of the payments made by all voters in an election. If we have less than a million voters, then the total payment will be less than 2^{40}. A discrete logarithm of this size can be extracted in less than a second, e.g., using a baby-step giant-step algorithm. We define the bound for our range proofs $n = \lceil \log b \rceil$, that is, 20 in our example above.

Each voter publishes a commitment $c_b = \mathsf{Com}_{\mathsf{cpk}}(b; r_b) = g_1^b h_1^{r_b}$ on his budget b together with a proof of validity of this commitment $\pi_{op}^b(c_b)$, and saves the opening r_b.

3.2 Voting

Ballot preparation. A voter who wishes to submit a vote of value v computes two ciphertexts $c = (d, c_1, c_2) \leftarrow \mathsf{Enc}_{\mathsf{pk}}(v)$ and $\hat{c} = (\hat{d}, \hat{c}_1, \hat{c}_2) \leftarrow \mathsf{Enc}_{\mathsf{pk}}(v^2)$ and proofs π_{va} and $\hat{\pi}_{va}$ of the validity of these ciphertexts.

The commitments d and $\hat{d}$ derived from these ciphertexts and a proof $\pi_{sq}(d, \hat{d})$ is computed in order to prove that the right payment is committed to. Eventually, we need to make sure that v lies in a proper range, that is, in $[-2^{n/2}, 2^{n/2}]$. This can be done by computing a proof $\pi_r^{n/2+1}(dg^{n/2})$. Note that if only positive votes are allowed the factor $g^{n/2}$ is simply left out.

In the case of several vote questions we need to repeat this process for each question.

Budget update. The voter then updates his budget as $c_b := c_b/\hat{d}$, and also records the updated opening $r_b := r_b - \hat{r}_d$, where $\hat{r}_d$ is the random exponent that was used to compute $\hat{d} = g_1^{v^2} h_1^{\hat{r}_d}$.

He then produces a proof that this budget is still positive, which can be done by computing a range proof $\hat{\pi}_r^n(c_b)$. For several vote questions we only need to do this once after having updated the budget with the vote payments for each question.

We may wonder why both proofs π_r and $\hat{\pi}_r$ are needed: the π_{op}^b and $\hat{\pi}_r$ proofs show that the initial budget is less than 2^n and that the payment v^2 that is made is less than b. However, the proof π_{sq} only proves that v^2 is a square of v in Z_q. As a result, v^2 could actually be any quadratic residue modulo q, which opens to many undesirable values thanks to modulo reduction. Showing in π_r that v is actually with the $[-2^{n/2}, 2^{n/2}]$ range makes sure that no reduction happens during the squaring.

Still, the square proof π_{sq} conveniently handles the case of negative votes: both roots of v^2 are valid witnesses.

Ballot preparation audit. The ballot preparation system of the voters is expected to provide a verification mechanism for this process. A traditional solution is to use a so-called Benaloh challenge [4]: the ballot preparation system commits to the voter on the value of these ciphertexts and proofs, e.g., by displaying a hash of all these values. The voter can then decide to challenge the ballot preparation system who then needs to release all the randomness that it used to prepare the ballot, which allows verifying the commitment on an independent device.

Ballot submission When the voter finished to challenge his ballot preparation device, he sends his vote $(c, \hat{c}, \pi_{va}, \hat{\pi}_{va}, \pi_{sq}, \pi_r, \hat{\pi}_r)$ to the Election Authority.

The Election Authority verifies π_{va} and $\hat{\pi}_{va}$, then publishes the vote audit data $(\mathsf{DCom}(c), \mathsf{DCom}(\hat{c}), \pi_{sq}, \pi_r, \hat{\pi}_r)$ next to the name of the voter on the public bulletin board. (Publishing the names makes it possible to verify who voted, e.g., by interrogating the voters, and removes the need to trust an Election Authority, or any other entity, for voter authentication – even if such authentication can remain useful to protect from ballot flooding).

The publication of $\mathsf{DCom}(\hat{c})$ is accompanied by an update of the commitment c_b on the budget available for the voter, which is publicly recomputed as $c_b := c_b / \mathsf{DCom}(\hat{c})$ and posted on the bulletin board.

3.3 Election Tally

Computing the election results The PPATs ciphertexts are homomorphically additive. So, multiplying the first series of PPATS cipehertexts, i.e., the c's together and decrypting the result yields the sum of the votes.

This decryption process can be made publicly verifiable without any additional proof: the authorities just need to publish the opening on the product of the d commitments of all voters: correctness follows from the binding property of the commitment scheme.

Budget updates. The same homomorphic addition can be performed on the second series of ciphertexts, i.e., the $\hat{c}$, which, after decryption, reveals the total amount spent during the election. This amount is posted on the bulletin board, together with an opening of the product of the $\hat{d}$ commitments of the voters.

That amount can now be equally split among the voters as a sum b_u per voter, and all the budget commitments on the board are then updated as $c_b := c_b g_2^{b_u}$. Voters can verify their updated budget, and keep making new proofs based on it, as the update process does not change the randomness of c_b, which the voter knows.

3.4 Election Audit

The various steps of the protocol can be verified in the natural way.

Parameter generation and initialization. The auditor verifies that pp have been produced according to the expected security parameter, and possibly verify the process of the generation of h_1 (in order to avoid the risks of a trapdoor).

The auditor also verifies that the right budget has been announced, and that the budget commitments c_b posted by the voters come with valid proofs π^b_{op}.

Vote validity. The auditor then verifies the validity of the π_{sq} and π_r proofs associated to each vote, and their uniqueness on the board. He verifies that each vote is associated to a legitimate voter, and questions voters (whether they are reported to have voted or not) to check that they agree with what is posted on the bulletin board on their behalf.

Tally validity. The auditor then computes the product of the d and $\hat{d}$ commitments in all the valid votes, and verifies that the Tally Tellers published a result and total election payments that is an opening of these commitment products.

Budget verification. The auditor recomputes the value of budget redistribution b_u and that all individual voter budgets have been updated accordingly.

3.5 Protocol efficiency

Most of the computational cost of our protocol lies in two steps: ballot preparation, and election tally. The setup cost (key generation, initial budget commitment) is essentially negligible (unless a large number of Tally Tellers is chosen, but we expect it to be more in the range of 3-5).

We make a rough estimate of these costs, focusing on the cost of the exponentiations in $\mathbb{G}_1$ and $\mathbb{G}_2$, and on the cost of multiplications when they come in a potentially large number compared to the exponentiations, that is, during the tally. We neglect the cost of computing hashes and of the arithmetic in $\mathbb{Z}_q$ in the NIZK proofs, which is expected to be smaller by a level of magnitude compared to the cost of the exponentiations that these proofs contain.

Our estimate gives an idea of the order of magnitude of the timings and of the practicality of our protocol. The exact performance will strongly depend on the actual arithmetic and cryptographic libraries that are used, and on the computing platform that is chosen.

Our timings are based on the benchmark of the PandA library of Chuengsatiansup et al. [9], and on the execution of the protocol on a single core of a 2012 Intel i5-3210M processor running at 2.5GHz. Their numbers are given in number of CPU cycles, which we convert into time based on the processor clock frequency.

Cost of ballot preparation. We consider a budget upper bound of $2^n = 2^{20}$ and a single choice question (which is the typical application case of quadratic voting). The operation count for the preparation of a ballot is available in Table 2. The resulting timing, based on the performance in Table 1, is then less than $8.4ms$.

	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_T$
Fixed base exponentiation	51 μs	135 μs	244 μs
Single multiplication	2.4 μs	6.4 μs	3 μs

Table 1. Cost of the main group operations

	$\mathbb{G}_1$	$\mathbb{G}_2$
c	1	3
$\hat{c}$	1	3
π_{va}	2	3
$\hat{\pi}_{va}$	2	3
π_{sq}	3	0
π_r	2n + 1	0
$\hat{\pi}_r$	4n + 1	0
π_{op}	1	0
Total:	$6n + 12$	12

Table 2. Count of exponentiations for ballot preparation.

Cost of the election tally. The bulk of the cost of the election tally will come from the verification of the validity of the individual ballots, which will be essentially the same as the cost of producing all the NIZK proofs. The marginal cost per ballot coming from the homomorphic addition of the votes and costs is indeed negligible: 1 multiplication in $\mathbb{G}_1$ and 2 multiplications in $\mathbb{G}_2$, that is, around $15\mu s$. The decryption operation has a cost that is constant and independent of the number of voters and will then be negligible as soon as we have a few thousand voters. The cost of the final discrete logarithm operation, needed to obtain the actual number of votes and election budget, will be around $\sqrt{m2^n}$ multiplications in $\mathbb{G}_T$ using the baby-step giant-step algorithm in an election with m ballots. If $m = 2^{20}$, we obtain a timing around 3 seconds.

4 Protocol analysis

We briefly discuss the security properties of the quadratic voting protocol presented in last section, their main assumptions and give arguments why the properties are satisfied.

4.1 Protocol correctness

The correctness of the protocol essentially follows from the additive homomorphic property of the PPATS encryption scheme and of Pedersen commitments.

As in traditional schemes based on homomorphic encryption, votes are encrypted, but now with their value that can be any integer (provided that the corresponding payment can be made). Tally tellers homomorphically add these votes and decrypt the election result.

The same happens with the encrypted quadratic payments. The balance of each voter is then adjusted twice: once after submission of the vote, then after redistribution of the election spendings.

4.2 Ballot privacy

The protocol offers computational privacy against the Election Authorities, provided that sufficiently many of them are honest (as defined by the threshold key generation protocol). These authorities receive ballots that are encrypted with a CPA secure encryption scheme, accompanied with various Σ-protocols that prove, among other things, the knowledge of the vote content.

This combination of encryption and proof of knowledge has been shown to lead to an NM-CPA non-malleable encryption scheme. This combination is known to be sufficient to offer ballot privacy when duplicate ballots are rejected [5].

Note that the amount of information revealed on the bulletin board is also very minimal, that is, we only reveal the total votes for each candidate/question and the total budget amount spent in the election.

4.3 Perfectly private audit trail

The protocol offers a perfectly private audit trail, or everlasting privacy in front of adversaries who can only access the election bulletin board. This follows from the fact that the only information posted by voters on the board is perfectly hiding commitments and perfect zero-knowledge proofs, and that the result of the election is posted as a simple opening of a perfectly hiding commitment on that result.

So, provided that the voters have access to good sources of randomness when they prepare their vote, the content of the board is simply statistically independent of each vote content.

Note that for usability, it might be better for the voters to hold only a single long-term key that can generate openings to their budgets via a pseudorandom generator, instead of having to update the key in each election. However, this would endanger the everlasting privacy.

4.4 Verifiability

Cast-as-intended verifiability The Benaloh challenge allows the voter to verify that the ballot preparation system prepares ciphertexts that match the voter intent.

Recorded-as-cast verifiability The bulletin board, assumed honest, displays the perfectly hiding part of the submitted ballot, which the voter can control to be correct. If it is correct, then the voter is guaranteed that his vote cannot be interpreted in an unexpected way, provided that the commitment scheme's binding property is not broken.

The Election Authorities are prevented from claiming that the vote is actually invalid due to an issue in the non-published part, because they are required to verify that validity (thanks to the corresponding π_{va} proofs) before publishing a ballot on the board. (Of course, authorities could also reject a valid ballot and not publish it, arguing that the voter transmitted it incorrectly. But this is the case of any verifiable remote voting scheme: the voter can only know that his vote will be taken into account after his vote is included on the bulletin board.)

Eligibility verifiability The bulletin board includes the name of every voter next to the ballot that it submitted. This is enough for an auditor to interrogate the voters, ask them whether they submitted a ballot or not and, if they did, ask them if it is accurately displayed on the bulletin board.

This mechanism protects form malicious authentication authorities who would submit votes on behalf of potential voters who would not pay attention to the election. The use of an authentication mechanism of course remains important in order to avoid that voters submit arbitrary ballots on behalf of arbitrary voters, which would simply result in declaring the election invalid as soon as it is observed.

Actually the eligibility verifiability is also strengthened by the extra budget structure compared to ordinary PPAT voting schemes. In order to vote, and pay for your vote, you need to hold an opening to your budget commitment. If a voter has already voted in an earlier election, this prevents ballot stuffing on their behalf. As an example, if the adversary somehow knows a voter will not be paying attention to the bulletin board e.g. being without internet connection for some time, then the adversary cannot abuse this and vote on his behalf. For first time voters we can, however, not give such guarantees.

Budget verifiability The c_b commitments and π^b_{op} proofs make it possible for anyone to observe that every voter received his correct initial budget.

The update of the voter budget is publicly performed, using the spending amount $\mathsf{DCom}(\hat{c})$ committed to as part of the ballot. The proof $\hat{\pi}_r$ makes it possible to verify that the spending is within the correct range, and the proof π_r ensures that the actually paid amount is the square value of the vote, seen as integers, as mentioned in last section.

Tallied-as-recorded verifiability After verification of the ballots that need to be included in the tally, any auditor can multiply the vote commitments $\mathsf{DCom}(c)$ together and obtain a commitment on the election result. The Tally Tellers are able to open that commitment thanks to the openings that they received for each individual vote. The finding of any different opening would break the computational binding property of the commitment scheme, which relies on the DDH assumption.

Budget update verifiability The opening of the total spendings in an election can be verified just as the vote tally. From this, the voter refund can be recomputed, and the updated commitments of every voter budget c_b can also be recomputed.

5 Conclusion and Outlook

In this paper we have presented an efficient protocol for electronic quadratic voting with everlasting privacy and end-to-end verifiability. The protocol uses perfectly hiding commitments for both vote choices and budgets to create a perfectly private audit trail. The constructions also facilitates easy threshold sharing of the secret election. In total, we have improved many aspects of the earlier protocol suggested in [17].

As it stands we don't allow transfer between different voters' budgets. This could easily be changed, but we think that both the everlasting privacy, universal verifiability and non-coupling to real currencies is an advantage over solutions using anonymous cryptocurrencies such as Zerocash. Note that allowing budget transfers also opens up to strategic voting since it would be more favorable to have equal-sized budgets when voting [17].

An improvement of the scheme would be to achieve receipt-freeness. In the present scheme, the commitments and corresponding openings could be used directly to prove to a vote-buyer that you voted according to his instructions. It is an important piece of future work to improve on this situation. If we allow budget transfers, it would maybe also impede strategic voting since you cannot get proof that the budget you give away will be used according to your preference. Note that whereas the receipt-freeness of the vote choice can follow similar ideas in other e-voting schemes, the budget is less straight-forward since in our construction we hold a key to unlock the budget. Especially to prevent forced-abstention attacks, it will be necessary to hide the budget from the coercer.

6 Acknowledgements

The authors acknowledge support from the Luxembourg National Research Fund (FNR) and Belgium Fonds de la Recherche Scientifique for the joint FNR/F.R.S.-FNRS project SeVoTe. PBR also acknowledges the FNR INTER project VoteVerif. This work has also been funded in part by the European Union (EU) and the Walloon Region through the FEDER project USERMedia (convention number 501907-379156).

References

1. Ben Adida, Olivier De Marneffe, Olivier Pereira, and Jean-Jacques Quisquater. Electing a university president using open-audit voting: Analysis of real-world use of helios. In *Proceedings of the 2009 Conference on Electronic Voting Technology/Workshop on Trustworthy Elections*, EVT/WOTE'09, pages 10–10, Berkeley, CA, USA, 2009. USENIX Association.
2. P. S. L. M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography – SAC'2005*, volume 3897 of *LNCS*, pages 319–331. Springer, 2006.

3. Mihir Bellare and Shafi Goldwasser. Verifiable partial key escrow. In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 78–91. ACM, 1997.
4. Josh Benaloh. Ballot casting assurance via voter-initiated poll station auditing. In *USENIX/ACCURATE Electronic Voting Technology Workshop, EVT'07*. USENIX Association, 2007.
5. David Bernhard, Olivier Pereira, and Bogdan Warinschi. How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios. In X. Wang and K. Sako, editors, *ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 626–643. Springer, 12 2012.
6. Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Proceedings of the Second International Conference on Theory of Cryptography*, TCC'05, pages 325–341, Berlin, Heidelberg, 2005. Springer-Verlag.
7. Jan Camenisch. *Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem*. PhD thesis, ETH Zurich, 1998.
8. Bharat Chandar and E Glen Weyl. Quadratic voting in finite populations. 2017.
9. Chitchanok Chuengsatiansup, Michael Naehrig, Pance Ribarski, and Peter Schwabe. Panda: Pairings and arithmetic. In *Pairing-Based Cryptography - Pairing 2013*, volume 8365 of *Lecture Notes in Computer Science*, pages 229–250. Springer, 2013.
10. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology - CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 1994.
11. Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Advances in Cryptology - EUROCRYPT '97*, volume 1233 of *LNCS*, pages 103–118. Springer, 1997.
12. Édouard Cuvelier, Olivier Pereira, and Thomas Peters. Election Verifiability or Ballot Privacy: Do We Need to Choose? In *European Symposium on Research in Computer Security – ESORICS 2013*, volume 8134 of *Lecture Notes in Computer Science*, pages 481–498. Springer, 9 2013.
13. Ivan Damgård. On sigma protocols, 2010. http://www.daimi.au.dk/ ivan/Sigma.pdf.
14. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.
15. Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *J. Cryptol.*, 20(1):51–83, January 2007.
16. Steven P. Lalley and E. Glen Weyl. Nash equilibria for a quadratic voting game. *CoRR*, abs/1409.0264, 2014.
17. Sunoo Park and Ronald L. Rivest. Towards secure quadratic voting. *Public Choice*, 172(1-2):151–175, 2017. https://eprint.iacr.org/2016/400.
18. Torben Pryds Pedersen. A threshold cryptosystem without a trusted party. In *Proceedings of the 10th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'91, pages 522–526, Berlin, Heidelberg, 1991. Springer-Verlag.
19. David Quarfoot, Douglas von Kohorn, Kevin Slavin, Rory Sutherland, David Goldstein, and Ellen Konar. Quadratic voting in the wild: real people, real votes. *Public Choice*, 172(1-2):283–303, 2017.

20. Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, SP '14, pages 459–474, Washington, DC, USA, 2014. IEEE Computer Society.
21. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology - CRYPTO '89*, volume 435, pages 239–252. Springer, 1989.
22. Douglas Wikström. Simplified submission of inputs to protocols. In *Security and Cryptography for Networks, 6th International Conference, SCN 2008*, volume 5229 of *LNCS*, pages 293–308. Springer, 2008.