PQKryvos: Post-Quantum Secure E-Voting With Flexible Ballot Formats and Public Tally-Hiding

Nicolas Huber, Pascal Reisert, and Ralf Küsters

Institute of Information Security
University of Stuttgart
firstname.secondname@sec.uni-stuttgart.de

Abstract. Fair and free elections are the foundation of democracies and democratic processes. They require voting protocols that guarantee the integrity and verifiability of the result, as well as the private choice of each voter. Currently deployed e-voting protocols rely on traditional hardness assumptions, like the discrete logarithm problem, to provide these security guarantees. They are not post-quantum secure (pq-secure). While first proposals for pq-secure protocols exist, they are limited in the variety of voting scenarios they can support and/or in terms of efficiency.
In this work, we therefore propose PQKryvos, an efficient and flexible pq-secure homomorphic e-voting protocol that can be instantiated for a wide variety of election methods and ballot formats. Our construction efficiently combines homomorphic lattice-based commitments with hash-based general-purpose proofs (GPZKPs) to ensure ballot correctness. As a pq-secure instantiation of the Kryvos framework introduced by Huber et al. (CCS 2022), PQKryvos not only provides voter privacy and (public) verifiability of the result, but additionally allows for the stronger privacy notion of public tally-hiding. Public tally-hiding ensures that only the intended election result (such as the full vote count or only the winner) is publicly revealed, while no additional information is leaked. This further improves the privacy for both voters and election candidates.
PQKryvos is the first homomorphic pq-secure e-voting protocol to generically support arbitrary ballot formats and the first to provide public tally-hiding. Our implementation and evaluation of PQKryvos demonstrate that it achieves practical performance for diverse election schemes and outperforms the original pre-quantum Kryvos instantiation in some settings. Moreover, we demonstrate that by utilizing GPZKPs, existing pq-secure e-voting protocols can support additional ballot formats, can be enhanced in their tallying phase, and can be extended to publicly tally-hiding protocols.

1 Introduction

In recent years remote electronic voting (e-voting) systems have become more efficient and secure, which has led to a growing adoption in real world elections, including internal elections by companies, associations, and political organizations, as well as political elections at the municipal, regional, and state levels [1–4]. Various election methods are used in practice, ranging from simple YES/NO voting over multiple-choice methods to complex preferential voting methods, such as Borda count and Instant Runoff-Voting (IRV) used for political elections, e.g., in Australia and the United States [5, 6]. All modern remote e-voting protocols are supposed to guarantee verifiable, privacy-preserving elections.

Verifiability. E-voting systems, as all complex systems, are prone to programming errors, or even deliberate manipulation, which can lead to the announcement of wrong election results. Numerous cases of errors and problems with electronic voting protocols have been reported (see, e.g., [7–11]). A fundamental security property of e-voting protocols is, hence, *(end-to-end) verifiability*, i.e., that every voter and every external observer should be able to verify whether votes were counted correctly and whether the final election result indeed corresponds to the votes cast by eligible voters, even if all or most of the involved parties, including voting servers, are malicious. Verifiability has been studied in numerous research papers (for an overview, see e.g. [12]) and is by now explicitly required in many real-world elections [3, 13].

Privacy and Tally-Hiding. The second core security property of e-voting protocols is *privacy* or *ballot secrecy*: noone should learn how a particular voter V voted. Of course, absolute secrecy is not always achievable: For instance, if all but one voter V abstain, the published election result reveals V's choice. Formal

privacy definitions [14,15], therefore define privacy of an e-voting protocol by requiring that it does not reveal substantially more about individual votes than what is inevitably leaked by the election result.

This leakage can further be minimized by revealing only necessary information in the result: While many traditional e-voting protocols and essentially all paper-based voting protocols reveal the full tally consisting of all (aggregated) individual votes, it is instead often sufficient and even desired to only output, e.g., the winner of an election or the best n candidates. As noted in [16,17], keeping the full tally hidden is not just conceptually appealing but also aligned with existing practices: several organizations - among them ACM SIGs[1] and the German Informatics Society[2] - sometimes choose not to publish the full tally, but only the election winner(s) – e.g., to spare losing candidates the embarrassment of visibly low support. Apart from *embarassing losing candidates*, revealing the full tally can *introduce voter biases* in multi-stage elections, *weaken mandates* in close races, and *facilitate gerrymandering* by exposing district-level vote distributions. Moreover, it can also enable so-called *Italian attacks*, a form of coercion in preferential elections: A ranking of multiple candidates in a preferential voting scheme can be essentially unique. In an Italian attack, a coercer uses this fact and instructs a voter to cast a highly unusual ranking. The coercer can then later verify compliance by checking whether that particular ranking appears in the tally.

Verifiable election protocols [16–19] that do not (necessarily) reveal the full tally and, hence, address the above issues are called *tally-hiding*. Some of these protocols [17–19] hide the full tally $\mathbb{T}$ from everyone, including election officials, and are therefore called *fully tally-hiding*, others [16] only prevent the public, e.g., voters and external observers, from seeing the full tally and are therefore called *publicly tally-hiding*. Note that public tally-hiding is therefore a weaker form of tally-hiding. Our new protocol, PQKryvos, will be publicly tally-hiding and therefore, compared to most protocols that only provide privacy, achieves a significantly stronger security property. We emphasize that in all currently proposed (fully or publicly) tally-hiding protocols, tally-hiding is an optional enhancement rather than a restriction. This will likewise be true for PQKryvos. That is, in cases, where revealing the full aggregated tally *is* desired (e.g., to collect detailed voting statistics), this can still be done without affecting any other privacy or verifiability properties of the protocol.

Post-Quantum Security. The security (i.e., verifiability, privacy, and tally-hiding) of almost all modern remote e-voting protocols, especially including all deployed ones, relies on the hardness of problems that can be solved efficiently by powerful quantum computers – e.g., on the discrete logarithm problem.

Hence, developing post-quantum secure (pq-secure) alternatives is a pressing issue, especially given the sensitivity and long-term importance of election data. While significant efforts have been made towards extending the cryptographic toolbox with pq-secure alternatives [20], only a few pq-secure remote e-voting protocols have been proposed so far. There are essentially three types of remote e-voting protocols that provide pq-secure elections, namely mix-net-based protocols, protocols that provide everlasting privacy, and homomorphic e-voting protocols. Classical (non-pq-secure) remote election protocols used in practice are either of the first or last kind. Here, we concentrate on homomorphic voting, but briefly mention relevant proposals of pq-secure protocols of the two other types of protocols as well.

Mix-Nets. In mix-net-based protocols, cast encrypted ballots are shuffled before decryption/counting by a sequence of mix servers. To make such protocols pq-secure, verifiable and pq-secure mix-nets have been proposed in the literature [21–28]. Compared to homomorphic approaches, mix-net-based approaches do not leverage aggregation, i.e., the decrypted tally consists of all anonymized, individual ballots in plain. This yields the advantage of being able to handle all kinds of ballot formats and not requiring voters to prove the validity of their cast encrypted ballots — invalid ballots can simply be removed after decryption. One drawback of mix-nets is that the computational cost of shuffle proofs grows with the number of voters. Moreover, while sometimes desired, mix-nets reveal the list of all (anonymized) individual votes, which may reveal more information about individual votes than necessary.

[1] See `https://www.acm.org/elections/sigs/election-results`.
[2] See `https://gi.de/ueber-uns/organisation/wahlen`, until and including 2020.

Everlasting Privacy. In such protocols, privacy does not rely on any (pre- or pq) hardness assumption but holds unconditionally against arbitrarily powerful attackers, including quantum ones (see [29] for an overview of the existing protocols). However, so far, none of these protocols addresses verifiability against quantum adversaries. Moreover, such protocols usually come without a formal security analysis and are often built on non-standard assumptions like the existence of an anonymous channel for each voter or a non-standard bulletin board (cf. [29]) – without explaining how they can be met or what implications regarding verifiability need to be considered. For example, as stated by Cuvelier et al. [30], their proposed approach for achieving everlasting privacy (perfectly private audit trail (PPAT)) introduces a private bulletin board for voters' ciphertexts alongside the usual public one. However, as observed by the authors, this creates a verifiability gap: talliers with access to the private board can falsely claim that, e.g., ciphertexts are malformed, and such claims cannot be verified by parties who only see the public board. Thus, introducing such bulletin board at least implies additional trust assumptions to preserve public verifiability.

Homomorphic Post-Quantum Secure E-Voting. As mentioned, here we follow the homomorphic approach. Some pq-secure homomorphic e-voting protocols have been proposed in the literature [1, 16, 17, 31–33]. They use (threshold) additively homomorphic encryption schemes E or additively homomorphic commitment schemes C to compute a full tally $\mathbb{T}$ by aggregating the ballots of the individual voters, and typically require a set of parties, called *talliers* or *trustees*, to compute and publish the election result.

More specifically, in homomorphic e-voting protocols based on homomorphic encryption, a voter produces a ballot $\boldsymbol{v}$, encrypts it to $\mathsf{Enc}(\boldsymbol{v})$ using a public key with the corresponding private key shared among the talliers, and publishes it; $\mathsf{Enc}(\boldsymbol{v})$ could also be a tuple of ciphertexts. By the homomorphic property of the encryption scheme, everyone (including the talliers) can then aggregate the published ciphertexts of the eligible voters to obtain the ciphertext $\mathsf{Enc}(\mathbb{T})$ of $\mathbb{T}$. A threshold of talliers then computes $f_{\mathsf{res}}(\mathbb{T})$ from $\mathsf{Enc}(\mathbb{T})$ and publishes $f_{\mathsf{res}}(\mathbb{T})$, where f_{res} is the result function. For example, a result function f_{res} can output only the candidate with the highest number of votes in $\mathbb{T}$. In homomorphic protocols without tally-hiding, f_{res} is the identity, and the talliers only perform distributed decryption. We note that, unlike mix-net-based approaches, homomorphic protocols do not reveal the entire list of (anonymized) individual votes: Even in the non-tally-hiding case, i.e., when choosing $f_{\mathsf{res}} = \mathsf{id}$, they reveal only the *aggregated* tally $\mathbb{T}$. Hence, they guarantee a higher level of vote privacy than mix-net-based approaches, but are not suitable for elections where revealing the entire list of individual votes is desired.

In order for this construction to produce a correct result $\mathsf{res} = f_{\mathsf{res}}(\mathbb{T})$, voters should be forced to only cast valid votes $\boldsymbol{v}$, e.g., if voters are allowed one vote each, they should not be able to assign two votes (to one or two different candidates). Additionally, the talliers need to prove that their output is in fact $f_{\mathsf{res}}(\mathbb{T})$.[3] Usually, these properties are guaranteed by zero-knowledge proofs (ZKPs) (produced by the voters and the talliers respectively). Homomorphic e-voting protocols based on homomorphic commitments work similarly (see below for details). Practically deployed homomorphic e-voting protocols usually employ (variants of) ElGamal encryption and Pedersen commitments, both of which are not pq-secure. Additionally, the used ZKPs often rely on traditional hardness assumptions, e.g., variants of disjunctive Chaum-Pedersen proofs for YES/NO voting or single choice [1].[4]

The most prominent proposals for pq-secure homomorphic e-voting use the additively homomorphic lattice-based commitment scheme by Baum et. al. (BDLOP) [35]: EVOLVE [32] and Epoque [33]. While they reach good efficiency for simple YES/NO votes, they cannot be used for other election formats. Similarly, other pq-secure e-voting protocols only work with one specific ballot format, e.g., [36] only supports Multi-Vote elections, where a voter can distribute multiple votes among the candidates.

Flexible Ballot Formats and Tally-Hiding with GPZKPs.

The restriction to very specific ballot formats is not only typical for pq-secure homomorphic e-voting protocols, but also common for non-pq-secure ones. In fact, most currently deployed homomorphic e-voting protocols use specifically tailored ZKPs. While such specialized ZKPs yield highly efficient protocols, adapting

[3] For brevity of presentation, we do not discuss other common checks here, like eligibility checks. We refer to [1, 16].

[4] In principle, ballot validity checks can also be performed using fully-homomorphic (threshold) encryption schemes (FHE) [34], but they might suffer from the low efficiency of current FHE schemes; and only support single-vote elections.

to different (potentially novel) voting methods is non-trivial. Although solutions have been proposed for various practically used voting methods [19,37,38], these solutions remain inflexible and even subtle changes to the set of valid ballots may necessitate severe redesign of the underlying ZKPs. For instance, the ZKP proposed in [38] for ranking-based ballots works for the standard case of Borda count, where all candidates are assigned a unique rank, but the adaptation to allow incomplete rankings is already non-trivial [39].

With their system Kryvos, Huber et al. [16, 40, 41] proposed using general-purpose zero-knowledge proof systems (GPZKPs) to flexibly support different ballot formats and to enable the talliers to prove in zero-knowledge the correct evaluation of a result function f_{res} on the full tally $\mathbb{T}$, yielding public tally-hiding. We follow this approach for our protocol PQKryvos, which can be seen as a pq-secure instantiation of the generic Kryvos framework.

1.1 Our Contributions

In this work, we propose PQKryvos, the first homomorphic pq-secure e-voting protocol that supports *essentially arbitrary ballot formats*, with *competitive efficiency in the pq-setting*. It is also the *first pq-secure e-voting protocol with public tally-hiding*. In a nutshell, PQKryvos works as follows: A voter V_i uses a pq-secure homomorphic commitment scheme to commit to the shares $\boldsymbol{v}_j^i$ of an additive full-threshold sharing of her vote $\boldsymbol{v}^i = \sum_j \boldsymbol{v}_j^i$ (one share per tallier). The voter then uses a pq-secure GPZKP to prove in zero-knowledge that the commitments have been correctly computed for shares of a valid vote. The voter publishes her commitments c_j^i and the ZKP. Additionally, each voter sends a ciphertext ct_j^i to each tallier T_j, encrypted with the tallier's respective public key using a pq-secure encryption scheme. ct_j^i contains the opening values for the tallier's respective commitment c_j^i. The talliers then jointly reconstruct an opening for the aggregated commitment $\sum_j \sum_i c_j^i$. By the additive homomorphism and binding property of commitments, this opening is the aggregated election tally $\mathbb{T} = \sum_i \boldsymbol{v}^i$. Finally, one of the talliers computes the election result $\mathsf{res} = f_{\mathsf{res}}(\mathbb{T})$ and uses a pq-secure GPZKP to prove that this result computation has been executed correctly. For a detailed and generic description of the overall protocol, that is independent of the used GPZKP and homomorphic commitment, see Section 2.

BDLOP and Ligero. To instatiate PQKryvos we use the state-of-the-art pq-secure BDLOP commitment scheme. Its security is well-understood and relies on standard hardness assumptions from post-quantum cryptography, like the Module-Learning with Errors problem (M-LWE) or Module-Shortest Integer Solution problem (M-SIS). Moreover, it uses the same message space as state-of-the-art pq-secure encryption schemes [42, 43] needed for the homomorphic aggregation of encrypted ballots in PQKryvos. BDLOP, therefore, was naturally the best currently available choice.

In PQKryvos we combine BDLOP commitments and pq-secure GPZKPs. For PQKryvos we require a pq-secure GPZKPs with high prover efficiency to ensure that voters can generate GPZKPs for ballot-validity proofs quickly. We observed that the linear structure of BDLOP commitments can be leveraged highly efficiently with GPZKPs that use Rank-1 Constraint Systems (R1CSs) as an intermediate representation. Based on these considerations, we identified Ligero [44] as the most suitable pq-secure candidate.

While both BDLOP and Ligero are state-of-the-art and Ligero comes with highly efficient implemenations, a practical combination of the two is not straightforward, since the best implementations of Ligero require SNARK-friendly prime modulus for the message space, e.g., the BN254 prime p_{BN254} or the Goldilocks prime p_{gl}. These primes are a priori incompatible with BDLOP.

By only using parts of BDLOP and proving that these parts are sufficient for our e-voting protocol, we were able to lift this restriction and to instantiate BDLOP with p_{BN254} and p_{gl}. We show that, as a positive side-effect of using a GPZKP for ballot validity, we no longer require the (amortized) ZKPs in the tallying phase common in state-of-the-art pq-secure BDLOP-based protocols like EVOLVE [32] and Epoque [33]. Our tallying phase is therefore significantly faster. Additionally, our pq-secure proofs of ballot validity are flexible and can be instantiated for essentially arbitrary ballot formats, including all voting methods supported by the original Kryvos protocol: Single- and Multi-Vote, two variants of Borda Count, instant-runoff voting (IRV), Condorcet methods, and Majority Judgment, making PQKryvos the first homomorphic pq-secure system to do so.

Our Implementation. We have implemented our circuits in [45]. Our implementation extends the `circom` library [46] to support the BDLOP message space and commitment scheme for prime modulus compatible with common SNARK fields used by Ligero. Our implementation can be used to prove arbitrary linear or quadratic relations of committed values in these spaces. It can therefore be used, independently of PQKryvos, in future applications and research to create ZKPs for statements about values on the BDLOP message space and/or their commitments. We provide example code in [45] for demonstrating a linear relation between committed (still hidden) values.

We show that our implementation is efficient and practical by using it to provide extensive benchmarks for PQKryvos. Our evaluation demonstrates that PQKryvos achieves practical performance - in most cases, even outperforming the original, pre-quantum Kryvos protocol - while preserving its flexibility and tally-hiding properties and adding pq-security. Compared to the state-of-the-art homomorphic pq-secure protocols like EVOLVE and Epoque [32, 33], our proof generation of ballot validity is slower, but remains well within practical range, needing 1.5 s per voter, compared to 8.5 ms in [32, 33]; a difference voters when casting their votes will likely not even recognize. Conversely, PQKryvos has the aforementioned optimized tallying phase, supports far more ballot formats, and additionally provides public tally-hiding. Finally, we remark that EVOLVE and Epoque could also be extended to tally-hiding protocols following our approach.

In summary, our paper makes the following contributions:

- Our new e-voting protocol PQKryvos supports essentially arbitrary ballot formats and election formats, including but not limited to single- and multiple-choice, Borda Count, instant-runoff voting (IRV), Condorcet methods, and Majority Judgment.
- PQKryvos is the first pq-secure e-voting scheme that supports public tally hiding, therewith improving the privacy of voters and candidates.
- PQKryvos shows that the pq-secure BDLOP commitments of [35] can be initiated with SNARK-friendly primes and can be used efficiently with Zero-Knowledge Succinct Non-Interactive Argument of Knowledges (zkSNARKs) like Ligero [44].
- Moreover, by using GPZKPs, our tallying phase of PQKryvos becomes less complex and hence faster than the tallying phase of state-of-the-art pq-secure protocols like [32, 33]. Additionally, we show in Section 4.2 that [32, 33] can be extended to support public tally-hiding.
- We have implemented our circuits [45] using `Circom` [46] and Ligero. Our implementation allows to construct constraint systems for arbitrary quadratic relations on the BDLOP message space and to transform them into ZKPs. Thereby, our implementation can serve as a building-block for any implementation that uses BDLOP-related computations, including but not limited to proofs of knowledge of a valid opening and linear relations between multiple commitments.
- Our benchmarks in Section 5 show that PQKryvos achieves practical performance for e-voting tasks, outperforming the original pre-quantum Kryvos protocol in almost all cases.

2 Secure E-Voting from GPZKPs

As outlined, our new protocol, PQKryvos, uses the same high-level framework as Kryvos [16], which is, however, non-pq-secure. We first briefly recall this common framework, its features, and the deficiencies of the currently existing (non-pq-secure) instantiation.

2.1 Overview of the High-Level Framework

Like Kryvos, PQKryvos is executed among the following set of participants: A voting authority Auth, n_v voters $\mathsf{V}_1, \ldots, \mathsf{V}_{n_v}$, n_t talliers $\mathsf{T}_1, \ldots, \mathsf{T}_{n_t}$ and a public bulletin board BB. We assume that a mutually authenticated channel to BB exists for each party.

A valid vote is encoded as an element $\boldsymbol{v} \in \mathsf{Ch} \subseteq \mathbb{F}_q^{n_{\mathsf{choices}}}$, where $\mathbb{F}_q$ is some finite field and Ch is a predefined set of valid votes, called a *choice space*. For instance, in a single-vote election with three candidates, where each voter can give a single vote to one out of three candidates, the choice space is given as $\mathsf{Ch}_{\mathsf{single}} =$

$\{\boldsymbol{x} \in \mathbb{F}_q^3 \mid x_1, x_2, x_3 \in \{0,1\} \wedge x_1 + x_2 + x_3 = 1\}$. We describe more complex choice spaces in Appendix B. Our framework is further characterized by a result function f_{res}. Based on the aggregated tally $\mathbb{T}$ (consisting of the total number of valid votes for each candidate), f_{res} computes the election result res. For instance, for $\mathsf{Ch}_{\mathsf{single}}$ from above, f_{res} could be defined to only return the candidate(s) that received the most votes.

We remark that while the design rationale of Kryvos is driven by the goal of designing a (publicly) tally-hiding e-voting protocol, it can be instantiated completely without a tally-hiding function f_{res}, or more formally, with $f_{\mathsf{res}} = \mathrm{id}$ set to trivially output the aggregated tally; the same will hold for PQKryvos.

Our common framework makes use of the following cryptographic primitives:

- An additively homomorphic binding and hiding commitment scheme $\mathsf{C} = \mathsf{C}_{\mathbb{F}_q^N} = (\mathsf{Setup}_{\mathsf{C}}, \mathsf{Com})$ that allows for commiting to elements of $\mathbb{F}_q^N$ for some $N > 0$.
- An IND-CCA2-secure public-key encryption scheme $\mathsf{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ encrypting values from $\mathbb{F}_q^N$ with verifiable decryption, i.e., with a protocol to compute a non-interactive zero-knowledge proof (NIZKP) π_{dec} for the relation $\mathfrak{R}_{\mathsf{dec}} = \{((x, \mathsf{c}, \mathsf{pk}); \mathsf{sk}) \mid (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen} \wedge \mathsf{Dec}_{\mathsf{sk}}(\mathsf{c}) = x\}$.
- A GPZKP $\Pi = (\mathsf{Setup}_{\Pi}, \mathsf{Prove}_{\Pi}, \mathsf{Verify}_{\Pi})$ that, given any binary relation $\mathfrak{R}$, enables the generation of a non-interactive zero-knowledge proof of knowledge (NIZKPoK) of a witness w for a public input x, whenever $(x, w) \in \mathfrak{R}$.

We get the following three phases of the overall protocol:

The Setup Phase. In the setup phase, Auth runs $\mathsf{Setup}_{\mathsf{C}}$ to determine the public parameters PP_{C} of the commitment scheme $\mathsf{C}_{\mathbb{F}_q^N}$ and publishes them on the BB. Moreover, Auth determines the list $\mathbf{id}$ of eligible voters and fixes the election method given by $\mathsf{Ch} \subseteq \mathbb{F}_q^{n_{\mathsf{choices}}}$ and the result function f_{res} (if applicable). We assume that $n_{\mathsf{choices}} = N$, which for smaller choice spaces can be achieved by padding valid votes with zeroes.[5] Auth publishes $\mathbf{id}, \mathsf{Ch}$, and f_{res} (if applicable) on BB. Furthermore, Auth runs Setup_{Π} for the ballot validity relation $\mathfrak{R}_{\mathsf{ballot}} = \{(c; (\boldsymbol{v}, r)) \mid \mathsf{Com}(\boldsymbol{v}, r) = c \wedge \boldsymbol{v} \in \mathsf{Ch}\}$ and, in the tally-hiding case, for the result relation $\mathfrak{R}_{f_{\mathsf{res}}} = \{(\mathrm{res}, c); (r, \mathbb{T}) \mid \mathsf{Com}(\mathbb{T}; r) = c \wedge \mathrm{res} \leftarrow f_{\mathsf{res}}(\mathbb{T})\}$ and publishes the resulting public parameters $\mathsf{PP}_{\mathsf{ballot}}^{\Pi}, \mathsf{PP}_{\mathsf{res}}^{\Pi}$ on BB.

Then, each tallier $\mathsf{T}_k, 1 \leq k \leq n_t$, runs the key generation algorithm KeyGen of E to generate its encryption/decryption key pair $(\mathsf{pk}_k, \mathsf{sk}_k)$ and posts (k, pk_k) on BB.

The Voting Phase. A voter V^i chooses a vote $\boldsymbol{v}^i \in \mathsf{Ch}$ and computes a (component-wise) full-threshold secret sharing of $\boldsymbol{v}^i$ among all n_t talliers as $\left(\boldsymbol{v}_k^i\right)_{k=1}^{n_t}$ with $\sum_{k=1}^{n_t} \boldsymbol{v}_k^i = \boldsymbol{v}^i$ in $\mathbb{F}_q$. Then, for each tallier T_k, V^i computes a commitment $c_k^i \leftarrow \mathsf{Com}(\boldsymbol{v}_k^i; r_k^i)$, where r_k^i denotes the used random coins. In addition to that, using Prove_{Π} with public parameters $\mathsf{PP}_{\mathsf{ballot}}^{\Pi}$, V^i computes a NIZKPoK π_{ballot}^i showing that the sum $c^i = \sum_{k=1}^{n_t} c_k^i$ is a commitment to an element of Ch. Finally, for each tallier T_k, V^i computes a ciphertext $\mathsf{ct}_k^i \leftarrow \mathsf{Enc}(\mathsf{pk}_k; (\boldsymbol{v}_k^i; r_k^i))$ encrypting T_k's share of $\boldsymbol{v}^i$ as well as the random coins used for committing to it. To complete the voting process, V^i submits her ballot $\mathsf{b}^i = \left(i, \pi_{\mathsf{ballot}}^i, \left(c_k^i, \mathsf{ct}_k^i\right)_{1 \leq k \leq n_t}\right)$ to BB. The BB then checks that all values in b^i are well-formed, that $i \in \mathbf{id}$, that no ballot has been submitted by V^i before, that π_{ballot}^i is valid (using Verify_{Π} with public parameters $\mathsf{PP}_{\mathsf{ballot}}^{\Pi}$), and that no previously submitted vote contains any of the ciphertexts ct_k^i.[6] If all checks pass, BB appends b^i to the public list of ballots on BB. Whenever a new ballot b^i is appended on BB, each tallier T_k decrypts their respective ciphertext ct_k^i to obtain a tuple $(\widetilde{\boldsymbol{v}}_k^i; \widetilde{r}_k^i) \leftarrow \mathsf{Dec}(\mathsf{sk}_k; \mathsf{ct}_k^i)$. T_k checks whether $(\widetilde{\boldsymbol{v}}_k^i; \widetilde{r}_k^i)$ is a valid opening to c_k^i by computing $\widetilde{c}_k^i \leftarrow \mathsf{Com}(\widetilde{\boldsymbol{v}}_k^i; \widetilde{r}_k^i)$ and checking whether $\widetilde{c}_k^i = c_k^i$. If this is not the case, T_k publishes a complaint on BB consisting of the decryption result $(\widetilde{\boldsymbol{v}}_k^i; \widetilde{r}_k^i)$ as well as a zero-knowledge proof π_{dec} of correct decryption. The election authorities can then review the claim by the tallier, possibly exclude the ballot/voter, or ask the voter to provide a new, correct ballot.

[5] Note that for larger choice spaces – e.g., required for preferential elections, such as IRV – where $n_{\mathsf{choices}} > N$, we can pad n_{choices} to be divisible by N and then consider n_{choices}/N-tuples of elements from $\mathbb{F}_q^{n_{\mathsf{choices}}}$. For ease of presentation we restrict to the case $n_{\mathsf{choices}} \leq N$ here and refer to the original paper [16] for the general case.

[6] Note that, due to the IND-CCA2-security of E, this check prevents copying attacks.

The Tallying Phase. We describe the tallying phase in two sub-steps, namely *(i)* the *Aggregation Step* and, if a result function f_{res} has been specified, *(ii)* the *Tally-Hiding Result Computation Step*. While the latter is a defining feature of Kryvos, this separation helps illustrate that removing the result computation still yields a verifiable (non-tally-hiding) e-voting protocol.

At this point (if all ballots passed the checks) each tallier T_k holds the openings $(\boldsymbol{v}_k^i, r_k^i)$ to c_k^i for $i = 1, \ldots, n_{\mathsf{voters}}$. Now, in the *Aggregation Step*, each tallier T_k first locally computes $\boldsymbol{t}_k = \sum_{i=1}^{n_v} \boldsymbol{v}_k^i$ and $r_k = \sum_{i=1}^{n_v} r_k^i$. Note that $(\boldsymbol{t}_k, r_k)$ is an opening for the aggregated commitment $c_k := \sum_{i=1}^{n_v} c_k^i$. Next, the talliers internally share their local openings $(\boldsymbol{t}_k, r_k)_{k=1}^{n_t}$ to compute $\mathbb{T} = \sum_{k=1}^{n_t} t_k = \sum_{i=1}^{n_v} \boldsymbol{v}^i$ and $r = \sum_{k=1}^{n_t} r_k$, yielding an opening $(\mathbb{T}, r)$ for the aggregated commitment $c = \sum_{i=1}^{n_v} \sum_{k=1}^{n_t} c_k^i$.

Since c can be computed by any party from public values on BB, the talliers can now publish $(\mathbb{T}, r)$ and anybody can check that it matches the voters' commitments. By construction, $\mathbb{T}$ is exactly the aggregated tally, i.e. the sum of all valid votes. This completes a non-tally-hiding instantiation of Kryvos or PQKryvos.

In a publicly tally-hiding instantiation, the talliers still compute $(\mathbb{T}, r)$ internally, but do not publish it. Instead, they execute the *Tally-Hiding Result Computation Step*. Here, one of the n_t talliers, called the *designated tallier*, computes the election result $\mathsf{res} \leftarrow f_{\mathsf{res}}(\mathbb{T})$. Then, using Prove_{Π} with public parameters $\mathsf{PP}_{\mathsf{res}}^{\Pi}$, the designated tallier computes a NIZKPoK π_{result} for $\mathfrak{R}_{f_{\mathsf{res}}}$ showing that res is obtained by applying f_{res} to a vector $\mathbb{T}$ for which random coins r exist, such that $c = \mathsf{Com}(\mathbb{T}, r)$. The designated tallier then publishes res and π_{result} on BB, allowing any party to verify that res has been computed correctly by applying Verify_{Π} with public parameters $\mathsf{PP}_{\mathsf{res}}^{\Pi}$.

2.2 The Original Kryvos Instantiation

In order to apply the above framework for concrete elections, the cryptographic primitives – commitment, encryption, GPZKPs – need to be instantiated appropriately and efficiently by finding suitable GPZKPs and circuits for the relations $\mathfrak{R}_{\mathsf{ballot}}$ (ballot validity relation) and $\mathfrak{R}_{f_{\mathsf{res}}}$ (result relation). Achieving this in a way that works well for a large class of ballot formats and result functions is the most challenging part. Importantly, the GPZKPs also need to work well with the chosen commitment scheme, as computing commitments is part of the circuits for which GPZKPs are performed; in fact, this generally makes up the biggest part of the circuits.

For Kryvos, Huber et al. use Pedersen commitments to instantiate C and Groth16 zkSNARKs [47] as their GPZKPs and build arithmetic circuits (resp. R1CSs) for a variety of choice spaces and result functions. However, since both Pedersen commitments and Groth16 zkSNARKs do not provide pq-security, this instantiation of Kryvos is not pq-secure. Moreover, the use of Groth16 zkSNARKs has two more shortcomings. Firstly, the security of Groth16 is based on elliptic curve pairings and it restricts $\mathbb{F}$ to be the scalar field of a pairing-friendly elliptic curve. Secondly, Groth16 requires an honestly generated, circuit-specific *common reference string* (CRS) as part of the public parameters $\mathsf{PP}_{\mathsf{ballot}}^{\Pi}$ and $\mathsf{PP}_{\mathsf{res}}^{\Pi}$. Each new instantiation of Kryvos with Groth16 for a different choice space or result function necessitates a trusted setup ceremony. In particular, if the honest generation of CRS is not ensured, the soundness of the derived NIZKPoKs π_{ballot} and π_{result} is no longer guaranteed. Huber et al. already observed this issue and discuss ways of distributing trust in the generation of CRS, but this introduces a threshold assumption for the verifiability of Kryvos, which is generally highly undesired for verifiability in the setting of e-voting (unlike for *vote privacy*, where threshold assumptions can often not be avoided).

Unlike this instantiation from [16], PQKryvos not only provides pq-security, but also overcomes the mentioned two shortcomings. For this purpose, we instantiate PQKryvos with a pq-secure and transparent GPZKP.[7] We identify Ligero [44] as a good candidate in Section 3 as, in addition to being pq-secure and transparent, it also does not put any restrictions on $\mathbb{F}$. In addition, we use the lattice-based BDLOP commitment scheme described in Section 4 and show how to instantiate it in a R1CS-friendly way to achieve pq-security. We demonstrate that, re-using the non-cryptography sub-circuits from [16], we can even outperform their (non-pq-secure) instantiation in terms of efficiency (cf. Section 5).

[7] A GPZKP is called *transparent* if it does not require a trusted setup.

3 Kryvos with a PQ-Secure ZKSNARK

We evaluated several candidate proof systems for instantiating Kryvos with a pq-secure GPZKP or, more precisely, a pq-secure zkSNARK. Currently, the research on pq-secure zkSNARKs follows two main directions: lattice-based proof systems [48–52] and purely hash-based proof systems [44, 53–56]. While the former is an active and fast-evolving area of research and lattice-based zkSNARKs are reaching performance levels that are increasingly competitive with established hash-based constructions, the latter are currently more mature regarding theoretical foundations and implementation support. For this reason, we focus our evaluation on hash-based zkSNARKs and consider a purely lattice-based instantiation of Kryvos interesting future work.

In terms of implementation support, it is practically advantageous to use a proof system that directly supports R1CSs. While not a theoretical requirement, R1CS currently is the most supported intermediate representation used for zkSNARKs and also allows us to directly re-use (parts of) the circuits from the original Kryvos implementation (and optimizations thereof), which were written for the R1CS-based Groth16 zkSNARK. This observation was already made in [40]. Moreover, as we illustrate in Section 5.2, R1CS allows for leveraging the structure of the commitments we consider for PQKryvos, yielding a highly efficient protocol. Restricting to R1CSs-based proof systems rules out constructions such as STARKs [54] (which use an intermediate representation called AIR) and PlonK-based constructions [57, 58], which are not directly comparable to R1CS-based constructions, for our purposes.

A second key requirement is prover efficiency, since in Kryvos each voter must use the zkSNARK to create a proof of ballot validity on the client side. Based on these conditions, we found the Ligero construction [44] to be the most suitable choice, as it offers a highly efficient prover and has been implemented directly for supporting R1CS. In addition to the aforementioned aspects, we note that - unlike the Groth16 zkSNARK used in the original instantiation of Kryvos - Ligero is transparent, i.e., it does not require a trusted setup to generate a common reference string. Hence, using Ligero, we can avoid the associated verifiability risks outlined in Section 2. Another practical advantage is the availability of `libiop` [59], which offers a full ready-to-use implementation of Ligero that belongs to the same code framework as the `libsnark` library used in the original Kryvos implementation [16]. Although `libiop` is no longer actively maintained, it provides a solid baseline for performance estimation, and we believe that more recent constructions, such as [60], could deliver even better results. Unfortunately, the current implementation [61] of [60] is not compatible with our approach yet. More precisely, it does (to the best of our knowledge) currently not support externally generated R1CSs. In particular, we do not see how the linear structure of the BDLOP commitments can be leveraged to gain efficiency with [61]. It remains, therefore, an open task for future research to integrate our approach with [61].

In the following, we briefly describe the Ligero protocol and recall key design choices and security results from [44].

3.1 Overview of Ligero

We give an overview of the Ligero protocol for proving R1CS satisfiability [44]. In the following, we describe the protocol as an *interactive oracle proof (IOP)* [62], that is, an interactive protocol where a prover $\mathcal{P}$ does not only send regular messages to a verifier $\mathcal{V}$, but can also grant oracle access to proof strings (consisting of evaluations of polynomials), i.e., $\mathcal{V}$ can query those at some positions.

Ben-Sasson et al. [62] showed that an IOP can be transformed into a non-interactive argument of knowledge by committing to the prover's oracle strings using Merkle trees and applying the Fiat-Shamir heuristic to remove interaction. Under certain conditions - which are proven to hold for Ligero [44] - the resulting non-interactive protocol's knowledge soundness error (in the random oracle model) essentially matches that of the underlying IOP up to a small factor determined by the number of random oracle queries and the security (collision resistance and output length) of the hash function used for the Merkle trees and the Fiat-Shamir transform. Moreover, if the underlying IOP is honest-verifier statistical zero-knowledge, the resulting non-interactive argument is statistical zero-knowledge.

Linear Codes. Ligero is parametrized by a *linear code* $\mathfrak{L} \subseteq \mathbb{F}^n$, where one typically uses $n = 2^\lambda$, and a randomized encoding function $\mathfrak{E} : \mathbb{F}^\ell \to \mathfrak{L}$. While the Ligero construction works for general linear codes

and various instantiations with various tradeoffs exist [56, 63], we restrict to the original case of $\mathfrak{L}$ being a Reed-Solomon code for PQKryvos as done in the `libiop` implementation. For this purpose, we denote by $\mathbb{F}[X]_k$ the vector space of all univariate polynomials over $\mathbb{F}$ of degree at most k. Then, for a fixed set $\eta = \{\eta_1, \ldots, \eta_n\} \subset \mathbb{F}$, called the *codeword domain*, we define $\mathfrak{L}$ as

$$\mathsf{RS}_{n,k}(\eta) := \left\{ \left(P(\eta_i)\right)_{i=1}^{n} \middle| P \in \mathbb{F}[X]_k \right\}.$$

In order to define an encoding function $\mathfrak{E}$ for this code, we fix another set $\zeta = \{\zeta_1, \ldots, \zeta_\ell\} \subset \mathbb{F}$ with $\ell < k$. We assume that $\zeta \cap \eta = \emptyset$ and call ζ the *systematic domain*. Then, to encode $\boldsymbol{x} = (x_1, \ldots, x_\ell) \in \mathbb{F}^\ell$, choose a random polynomial $P_{\boldsymbol{x}} \in \mathbb{F}[X]_k$ such that $P_{\boldsymbol{x}}(\zeta_i) = x_i$ and define $\mathfrak{E}(\boldsymbol{x}) := \left(P_{\boldsymbol{x}}(\eta_i)\right)_{i=1}^{n}$.

Finally, for a matrix N with rows $N[i] \in \mathbb{F}^\ell$, we denote by $\mathfrak{E}(N)$ the matrix with rows $\mathfrak{E}(N[i]) \in \mathbb{F}^n$.

The Ligero IOP. A prover $\mathcal{P}$ and a verifier $\mathcal{V}$ agree on an R1CS, i.e., a triplet (A, B, C) of matrices $A, B, C \in \mathbb{F}^{\mu \times (\nu+1)}$, where μ is the number of constraints and ν is the number of involved variables. As mentioned in Section 2.2, any arithmetic circuit can be expressed in this form. We assume $\mu = r_\mu \cdot \ell$ and $\nu + 1 = r_\nu \cdot \ell$ for some $r_\mu, r_\nu \in \mathbb{N}$.[8] The goal of the Ligero protocol is for $\mathcal{P}$ to prove knowledge of a vector $z \in \mathbb{F}^{\nu+1}$ such that $(Az) \odot (Bz) = Cz$, where $\odot$ denotes the entry-wise product.[9] Instead of using the classical form of an R1CS above, in Ligero we interpret $a = Az, b = Bz$, and $c = Cz$ as $(r_\mu \times \ell)$-matrices and z as $(r_\nu \times \ell)$-matrix over $\mathbb{F}$. The Ligero protocol is then decomposed into the following three subprotocols:

1. *Linear Check:* For $(h, H) \in \{(a, A), (b, B), (c, C)\}$ show that $h = Hz$ to a verifier with oracle access to the columns of $\mathfrak{E}(h)$ and $\mathfrak{E}(z)$.
2. *Quadratic Check:* Show that $a \odot b - c = 0$ to a verifier with oracle access to the columns of $\mathfrak{E}(a), \mathfrak{E}(b), \mathfrak{E}(c)$.
3. *Code Check:* For a matrix $M \in \mathbb{F}^{r \times n}$ (for some $r > 0$) show that $M = \mathfrak{E}(N)$ for some $N \in \mathbb{F}^{r \times \ell}$ to a verifier with oracle access to the columns of M.

The high-level idea, for all those checks, is that the provided encoding acts as a commitment from $\mathcal{P}$ to the polynomials used for encoding, i.e., after granting oracle access, $\mathcal{P}$ can no longer change the polynomials. $\mathcal{V}$ then asks $\mathcal{P}$ to provide a random linear combination of the polynomials used for encoding and checks that the claimed equation holds when evaluating the random linear combination at ζ. Moreover, to ensure correctness, $\mathcal{V}$ queries the oracle at a few randomly chosen evaluations in η to spot-check consistency with the claimed linear combination.

We restrict to the combined overall protocol for R1CS satisfiability in the following. In order to achieve (honest-verifier) zero-knowledge, all three checks should additionally ensure that $\mathcal{V}$ cannot learn anything about the encoded values. This can be achieved by adding blinding values to $\mathcal{P}$'s response polynomials. We refer to [44] for details.

The Overall Protocol The overall Ligero protocol combines (and partially parallelizes) the three checks as follows:

Commitment $\mathcal{P}$ computes the row-wise encodings $\mathfrak{E}(a), \mathfrak{E}(b), \mathfrak{E}(c)$, and $\mathfrak{E}(z)$ using polynomials $P_{a_i}, P_{b_i}, P_{c_i}$, and P_{z_i} and sets $M = \begin{pmatrix} \mathfrak{E}(a) \\ \mathfrak{E}(b) \\ \mathfrak{E}(c) \\ \mathfrak{E}(z) \end{pmatrix} \in \mathbb{F}^{r \times n}$ with $r = (3r_\mu + r_\nu)$. Then, $\mathcal{P}$ grants oracle access to the proof string π_M consisting of the $n = 2^\lambda$ columns of M.

Challenge $\mathcal{V}$ sends random $\rho^{code} \in \mathbb{F}^r$, $\rho^{\mathsf{quad}} \in \mathbb{F}^{r_\mu}$, and $\rho^{\mathsf{lin}} \in \mathbb{F}^\mu$.

Response For $(h, H) \in \{(a, A), (b, B), (c, C)\}$, $\mathcal{P}$ computes $M_H = \left(\sum_{i=1}^{\mu} \rho_i^{\mathsf{lin}} H_{ik}\right)_{k=1}^{r_\nu \cdot \ell} \in \mathbb{F}^{\nu+1}$, interprets M_H as a $r_\nu \times \ell$ matrix and chooses row-wise corresponding polynomials $P_{M_H[j]}$. Similarly, $\mathcal{P}$ chooses r_μ

[8] Note that we can pad A, B, C with zeroes, if necessary, until $\ell \mid \mu, \nu + 1$.

[9] Usually, z is assumed to have the value 1 in its first entry to avoid the zero vector from being a trivial solution. Moreover, one sometimes fixes some of the first entries to public values. For the ease of presentation, we describe the protocol for the case that z contains no public values. Ensuring a public prefix only requires minor modifications.

polynomials $P_{\rho^{\mathsf{lin}}[i]}$ corresponding to the rows of ρ^{lin} and sends

$$P_{\mathsf{lin},h} = \sum_{i=1}^{r_\mu} P_{\rho^{\mathsf{lin}}[i]} \cdot P_{h_i} - \sum_{j=1}^{r_\nu} P_{M_H[j]} \cdot P_{z_j} \tag{$*$}$$

$$P_{\mathsf{quad}} = \sum_{i=1}^{r_\mu} \rho_i^{\mathsf{quad}} \cdot (P_{a_i} \cdot P_{b_i} - P_{c_i}) \tag{$**$}$$

$$\boldsymbol{v} = \left(\rho^{code}\right)^\top \cdot M \in \mathbb{F}^n. \tag{$***$}$$

Querying/Verification $\mathcal{V}$ checks that $\sum_{i=1}^{\ell} P_{\mathsf{lin},h}(\zeta_i) = 0$ (for $h \in \{a, b, c\}$), that P_{quad} vanishes on ζ, and that $\boldsymbol{v} \in \mathsf{RS}_{n,k,n-k+1}(\eta)$. Finally, $\mathcal{V}$ queries $t < k - \ell$ random columns of M and checks that $(*)$, $(**)$ $(***)$ hold at the query points. The first three checks ensure that $h = Hz$, that $a \odot b - c = 0$, and that M is a well-formed matrix of codewords, respectively. The checks at the query points ensure that the sent linear combinations match the encoded polynomials.

3.2 Security of Ligero and Practical Considerations

Security. Ames et al. [44] prove that the Ligero zero-knowledge IOP, consisting of σ iterations of the blinded version of the protocol outlined above, achieves perfect completeness, perfect honest-verifier zero-knowledge, and a soundness error of $\delta_s \leq \frac{n-k+3}{|\mathbb{F}|^\sigma} + (1 - \frac{e}{n})^t + 2\left(\frac{e+2k}{n}\right)^t$, where $e < \frac{n-k}{4}$. Ames et al. further show that Ligero meets the conditions given in [62] for transforming the described IOP to a non-interactive argument of knowledge. Hence, committing to the proof strings using Merkle trees and removing interaction via the Fiat-Shamir heuristic yields a non-interactive zero-knowledge argument that achieves a concrete soundness error of roughly $2^{-\kappa}$ if $\delta_s = 2^{-\kappa}$ and the used hash function outputs strings of length 2κ bits. Regarding the security against quantum adversaries, Chiesa et al. [64] proved that the soundness results from [62] can be lifted to the quantum random oracle (QROM) model. However, to maintain security against Grover-type search attacks [65], which reduce the effective security of a hash function by a factor of up to $\frac{1}{2}$, one should choose a hash length of 4κ bits, yielding 512 bits for achieving $\kappa = 128$. We remark that fewer bits might be sufficient to guarantee 128-bits security, e.g., with a more detailed analysis of the effects of Grover-type attacks as in [66]. In practice, a concrete hash function must be chosen. E.g., per default, the libiop implementation uses blake2b with variable hash length, which we set to 512 bits for obtaining our benchmarks in Sections 5.2 and 5.3. Overall, these considerations yield:

Theorem 1 (Security of Ligero). *The blinded non-interactive Ligero protocol obtained from the blinded Ligero IOP by applying the transformation from [62] (i.e., replacing oracles with Merkle trees and removing interactivity with Fiat-Shamir) described above is a non-interactive GPZKP that, for a given R1CS (A, B, C) gives a non-interactive zero-knowlede proof of knowledge that, for $\delta_s \leq 2^{-\kappa}$ and a hash-length of 4κ bits achieves perfect zero-knowledge and a soundness error of $\sim 2^{-\kappa}$ in the QROM model.*

Practical Considerations. The theory of Ligero outlined above, allows, in principle, to use arbitrary fields $\mathbb{F}$ for expressing the arithmetic relation to be proven - unlike the Groth16 zkSNARK used in [16], which is limited to scalar fields of pairing-friendly elliptic curves such as BN254. Moreover, again unlike Groth16, note that Ligero requires no setup beyond the R1CS generation and, in particular, no honestly generated CRS. This not only allows for mitigating the verifiability issues of the Groth16 instantiation outlined in Section 2, but also allows for updating a previously generated R1CS, e.g., by including additional constraints, without having to re-run the entire generation process.

That being said, for practical deployment, one still needs to generate the R1CS used for a concrete relation at least once, which can be complex. Fortunately, due to its widespread adoption, several libraries for compiling arithmetic programs to R1CS exist, most notably `circom` [46]. While `circom` was designed with Groth16 in mind, its R1CS generation process can be used independently, making it suitable as a front-end compiler for Ligero, too. This, however, restricts compilation to the fields supported by `circom`. To

avoid re-inventing R1CS generation, we therefore restrict ourselves in the following to the fields supported by circom and concretely consider the scalar field of BN254 (its order being a 254-bits prime, which we denote by p_{BN254}) and the field $\mathbb{F}_{p_{\mathsf{gl}}}$ over the *Goldilocks* prime $p_{\mathsf{gl}} = 2^{64} - 2^{32} + 1$. By choosing one of these primes, we also ensure that we can profit from future optimization of popular implementations like `circom`. We emphasize that this is a practical consideration, not a theoretical limitation of the Ligero proof system.

4 Lattice-Based Commitments Over SNARK-Friendly Primes

In this section, we describe the commitment scheme used in our new protocol PQKryvos and discuss how to instantiate it efficiently and securely. We choose an additively homomorphic lattice-based commitment scheme, namely the BDLOP [35]. BDLOP is already used in pq-secure e-voting protocols like [32,33,36] but has not been used for tally-hiding e-voting or with GPZKPs.

Unfortunately, BDLOP puts very specific conditions on the order of the message space $\mathbb{F}_q$, which are not satisfied (as far as we know) by our SNARK-friendly primes p_{BN254} and p_{gl} (cf. Section 3). A simple solution to this incompatibility problem is to use different primes for the proof system and the commitment scheme. However, this *foreign field arithmetic* requires ensuring correctness of all operations modulo q inside a computation modulo a SNARK-friendly prime and imposes new constraints that increase the R1CS size and significantly impact performance. We also tested this approach for comparison, and its performance is still somewhat acceptable; see our discussion in Section 5. However, in this section, we prove that we can also instantiate a variant of BDLOP over our SNARK-friendly primes, completely avoiding the overhead from foreign field arithmetic. We believe that this result is of independent interest and may have applications beyond electronic voting.

Instead of using this straightforward instantiation that would yield expensive foreign field arithmetic, in PQKryvos, we extend and modify BDLOP to work with SNARK-friendly primes. We can then use the same (SNARK-friendly) prime for both the commitment scheme and the proof system. This promises a much smaller R1CS and, hence, a more efficient proof system. We explored two options in this direction, which, we discuss in Section 4.2 and Section 4.3 below, respectively. We start by giving a short recap of BDLOP.

4.1 Recap: The BDLOP Commitment Scheme

We recap the BDLOP commitment scheme in its simplified notation from [32] since it is sufficient for our applications.

Notation. Let q be an odd prime and m a non-trivial power of some other prime. Let deg be the order of q in $\mathbb{Z}_m^\times$. Define $K = \mathbb{Q}[X]/(\Phi_m)$ the cyclotomic extension with cyclotomic polynomial Φ_m of degree $N \coloneqq \phi(m)$ and $R = \mathbb{Z}[X]/(\Phi_m)$ its ring of integers. We denote $R_q \coloneqq R/(q) \simeq \mathbb{F}_q[X]/(\Phi_m)$. For $y = \sum_{i=0}^{N-1} y_i X^i \in R_q$ with representatives[10] $y_i \in \{(1-q)/2, \ldots, (q-1)/2\}$ we define $\|y\|_t = \|(y_0, \ldots, y_{N-1})\|_t$ for the t-norm of vectors, $1 \leq t \leq \infty$. Let d, B_r, β be positive public integers and $x \in R_q$ a message to commit to. Let $S_{t,\beta} \coloneqq \{y \in R_q : \|y\|_t \leq \beta\}$ and $\mathcal{C} = S_{\infty,1} \cap S_{1,60}$ the challenge space. Moreover, let $\overline{\mathcal{C}} = \{y - y' : y \neq y' \in \mathcal{C}\}$.

Then, the functionalities of the BDLOP commitment scheme $\mathsf{BDLOP}(q, N, d, \beta, B_r)$ are defined as follows:

KeyGen. Let $A_1' \leftarrow R_q^{d \times (d+1)}$, $A_2' \leftarrow R_q^{1 \times d}$. The public parameters of the commitment scheme are defined as $A_1 = (A_1', I_d) \in R_q^{d \times (2d+1)}$ and $A_2 = (A_2', 1, 0, \ldots, 0) \in R_q^{1 \times (2d+1)}$ where I_d denotes the $d \times d$ identity matrix. Denote $A = (A_1^T, A_2^T)^T \in R_q^{(d+1) \times (2d+1)}$.

Commit. To commit to $x \in R_q$, sample a random coin $r \leftarrow S_{\infty,\beta}^{2d+1}$ and compute the commitment as

$$\mathsf{Com}(x, r) = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \coloneqq Ar + \begin{pmatrix} 0 \\ x \end{pmatrix}. \tag{1}$$

[10] In our implementation [45] and Section 5 we use representatives in $\{0, \ldots, q-1\}$. For the theoretical considerations in this section we however chose the more common notion with coefficients centered around 0.

Open. A valid opening of a commitment $(c_1, c_2)^T$ is a pair $(x, r) \in R_q \times S_{2,B_r}^{2d+1}$ such that (1) holds. An approximate opening of $(c_1, c_2)^T$ is a triple $(x, r, f) \in R_q \times S_{2,B_r}^{2d+1} \times \overline{\mathcal{C}}$ such that

$$f \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = Ar + f \begin{pmatrix} 0 \\ x \end{pmatrix}. \tag{2}$$

We have the following security guarantees related to the hardness of the M-LWE and the M-SIS (cf. Definitions 1 and 2 in Appendix A). For the proofs of [32,35] with a slight adaptation to our setup see Appendix A.

Theorem 2. *The commitment scheme* $\mathsf{BDLOP}(q, N, d, \beta, B_r)$ *described above is computationally hiding if* $\mathsf{MLWE}(q, N, d, S_{\infty,\beta})$ *is computationally hard. It is computationally binding for the classical opening if* $\mathsf{MSIS}(q, N, d, 2d+1, 2B_r, A_1)$ *is computationally hard. If every element of* $\overline{\mathcal{C}}$ *is invertible, then it is computationally binding for the approximate opening if* $\mathsf{MSIS}(q, N, d, 2d+1, 4\sqrt{60}B_r, A_1)$ *is computationally hard.*

Remark 1. $\mathsf{BDLOP}(q, N, d, \beta, B_r)$ is additively homomorphic in the coefficients of polynomials in R_q. Since $R_q \simeq \mathbb{F}_q^N$ we can interpret $\mathsf{BDLOP}(q, N, d, \beta, B_r)$ as a homomorphic commitment scheme on $\mathbb{F}_q^N$. We will use this interpretation throughout our paper.

4.2 BDLOP for SNARK-Friendly Primes

Theorem 2 requires invertibility of elements in $\overline{\mathcal{C}}$. Since all elements of $\overline{\mathcal{C}}$ have an $\|\cdot\|_\infty-$norm of at most 2, the condition is satisfied if all sufficiently small elements in R_q are invertible. To guarantee the invertibility, [35] uses Corollary 1.2 of [67] which guarantees invertibility of elements of $\overline{\mathcal{C}}$, if *(i)* m is a power of two, *(ii)* $q = 1 + 2k \bmod 4k$ for $m > k > 1$, and *(iii)* $2^k\sqrt{2k} < q$. Choosing m a power of two leads to $\Phi_m(X) = X^N+1$ for $m > 2$ (with $N = m/2$). The modulo operation in R_q is therefore a simple (nega-cyclic) convolution and can be implemented more efficiently than for other choices of m (and has other nice algebraic features not relevant in our context).

A typical example of an admissible prime is $q = 2^{31} - 2^7 - 2^5 + 1$ used in [32], where one can choose $k = 2^4$ and $m > k$ and hence $2^k\sqrt{2k} = 2^{18.5} < q$. For the $\mathsf{BN254}$ prime p_{BN254}, we have that $2^{28} \mid (p_{\mathsf{BN254}} - 1)$ and for Goldilocks $2^{32} \mid (p_{\mathsf{gl}} - 1)$. Hence the smallest possible k with $q = 1 \bmod 2k$ is $k = 2^{27}$ for BN254 and $k = 2^{31}$ for Goldilocks. However, in both cases the inequality *(iii)* is unfortunately not satisfied. For this reason, we cannot directly work with a combination of power-two cyclotomics and p_{BN254} or p_{gl}. However, we can replace the condition $q = 1 + 2k \bmod 4k$ for the power-two case with an analogue for other prime powers due to Lyubashevsky and Seiler:

Theorem 3 (Theorem 1.1 of [67]). *Let p have order $deg \mid m$ in $\mathbb{Z}_m^\times$ and $z := m/deg$. If $q = 1 \bmod z$ and $0 \neq y \in R_q$ satisfies $\sqrt{z}\|y\|_\infty \leq p^{1/\phi(z)}$, then y is invertible.*

With this result, we can use BDLOP with powers of 3. Note that for elements of $\overline{\mathcal{C}}$ used in the approximate opening, the condition of Theorem 3 is satisfied if $2\sqrt{z} \leq p^{1/\phi(z)}$ and all elements of $\overline{\mathcal{C}}$ are invertible in this case. The latter condition is satisfied for p_{BN254} and $m = 3^j$ for $j \geq 2$, since $deg = 3^{j-2}, z = 9$ and $2\sqrt{z} = 6 \leq p^{1/\phi(z)} \approx 2^{254/6} \approx 2^{42.3}$. It is also satisfied for p_{gl} and $m = 3^j$ for $j \geq 1$, since $deg = 3^{j-1}, z = 3$ and $2\sqrt{3} < 4 \leq p^{1/\phi(z)} \approx 2^{64/4} = 2^{16}$. Hence, the proof of Theorem 2, e.g., in Appendix A, shows the security features of the corresponding instantiation of BDLOP.

Remark 2. Baum et al. [35] use the aforementioned condition $q = 1 + 2k \bmod 4k$, i.e. $z = 2k$ in Theorem 3. In addition to the invertibility property, this choice also ensures the existence of suitable primes and that their order mod m can be deduced from the order mod z theoretically. However, in our case, we already chose primes, namely p_{BN254} and p_{gl}, and can compute their order explicitly for a given m.

By extending the set of available primes for BDLOP, we can instantiate the full BDLOP scheme over a SNARK-friendly prime, enabling the integration of Ligero over the corresponding prime fields into BDLOP-based protocols where needed. For example, we can use the (unmodified) pq-secure e-voting protocol of [32] (and its end-to-end verifiable extension [33]) and extend it to a publicly tally-hiding protocol by adding a

result function f_{res} on the tally $\mathbb{T}$ and use Ligero to generate a ZKP π_{result} showing correct evaluation of f_{res} by the talliers. This yields a (publicly) tally-hiding pq-secure e-voting protocol. However, if we keep the specialized ZKPs of [32], we can still only handle YES/NO elections. If we replace them with Ligero SNARKs for computing π_{ballot} for more complex choice spaces, the resulting scheme can handle essentially arbitrary election formats. However, we will show below that by replacing the ballot validity proofs in [32], we no longer need approximate openings and can, therefore, omit the invertibility condition altogether.

4.3 Using a Simplified Version of BDLOP

PQ-secure e-voting protocols like [32] and its extensions use approximate openings in the BDLOP commentiment scheme for ballot validity ZKPs and the amortization of talliers' ZKPs. In PQKryvos we replace the former by GPZKPs, namely Ligero discussed in Section 3. In [32], the talliers' ZKPs are used to reduce the randomness inside an aggregated commitment, since increasing the randomness during aggregation might break the binding property and hence verifiability of the election scheme. We describe the construction of [32] in more detail in Appendix A.

While these ZKPs use the binding property of approximate openings, they only require the constant polynomial 2 to be invertible, which is always the case for an odd prime modulus, e.g., for p_{BN254} and p_{gl}. In particular, if we only want to use the talliers' ZKPs from [32], then we do not need to restrict the choice of m as in Theorem 3. Thus, we could choose m a power of 2 (given the general advantages of power-two cyclotomics we discussed above). However, as we explain in Section 5.2, the choice of m does not affect the overall number of constraints required for expressing the computation of a BDLOP commitment as R1CS, so choosing m a power of 2 yields no advantage in our setting. For our benchmarks, we chose a power-of-three that works for both approaches - Section 4.2, 4.3.

In principle, the talliers' ZKPs could be replaced by GPZKPs. However, in contrast to the ballot validity ZKPs in [32], the talliers' ZKPs can be used for an essentially arbitrary election system without modification – they are agnostic of the election method. Moreover, the aforementioned issues with too large random coins under aggregation, which would break the binding property, occur in [32] much earlier than in our setup with GPZKPs. The reason for this is that the special ZKPs used in [32] for YES/NO elections introduce a large slack, i.e., a proof only allows to extract a witness of possibly significantly larger size. Hence, a random coin's (provable) size inside an aggregated commitment increases fast. For our GPZKPs, we have a trivial slack of 1 instead. Thus, we only require talliers' ZKPs for extremely large elections. Concretely, for p_{gl}, the GPZKPs allow aggregation of up to 2^{61} ballots in PQKryvos without having to reduce the randomness and, hence, without requiring the randomness reduction ZKPs by the talliers. Even when using the prime $2^{31}-2^7-2^5+1$ of [32,33], this approach still supports 2^{28} ballots. In particular, if one replaces the specialized ballot-validity ZKPs in [32] with GPZKPs, then the talliers do not need to reduce the randomness size in almost all realistic elections, which can heavily improve the tallying efficiency of [32].[11] Note that in the standard version of [32] only around 100 voters are supported without randomness reduction for $N = \deg(\Phi_m) = 256, d = 7$, due to the large slack of the specialized ZKP for ballot validity in [32].

Overall, we saw in Section 4.2 that we can initiate [32] with a SNARK-friendly prime and then trivially extend it to a publicly tally-hiding e-voting protocol. In Section 4.3 we explained that by abandoning the ballot validity proofs of [32], we can use BDLOP without approximate opening and hence instantiate it with SNARK-friendly primes. Finally, by replacing ballot validity proofs of [32] with GPZKPs, the tallying phase becomes significantly more efficient as we do not need the (approximate) talliers' ZKPs from [32] for any realistic election scenario. Our PQKryvos protocol combines all of the above-mentioned advantages.

5 Instantiation and Implementation

In this section, we describe the overall PQKryvos protocol using the building blocks developed in Sections 3 and 4. We describe our design choices and how they affect the overall performance of PQKryvos. Finally,

[11] Note that apart from the not required talliers' ZKPs, PQKryvos shares a common tallying phase with [32] (cf. Section 2).

we evaluate PQKryvos based on our implementation [45] for the various choice spaces and result functions considered in [16] alongside a performance comparison.

5.1 The PQKryvos Protocol

As PQKryvos is an instantiation of the Kryvos framework, we refer mostly to our detailed recap from Section 2. Concretely, we instantiate Π with the Ligero GPZKP from Section 3 and the commitment scheme C with BDLOP commitments. We denote by $\mathbb{F}_q$ the field over which the BDLOP commitments are instantiated and by N the degree of the underlying cyclotomic polynomial. Since C is additively homomorphic in each of the N polynomial coefficients, we get $\mathsf{C} = \mathsf{C}_{\mathbb{F}_q^N}$. For the IND-CCA2-secure public-key encryption scheme E we use ML-KEM [43] together with the verifiable decryption from Lyubashevsky et al. [68]. We note that alternatively, one could follow the suggestion from [33] and use an adaptation of the identity-based encryption scheme from Agrawal et al. [69] or similar constructions [70, 71].

With these primitives, the overall protocol works exactly as described in Section 2. While the security properties of PQKryvos can be seen as an immediate corollary from the security analysis from [16], we provide an independent security analysis in Section 6 with additional details in Appendix C to ensure the exposition is self-contained. In the remainder of this section, we focus on the practical aspects of our instantiation and implementation.

5.2 Efficient Circuits for BDLOP Commitments

Recall from Section 2 that the main computational overhead in the original Kryvos instantiation stems from the cryptographic schemes expressed as constraints in the R1CS. We want to explain how the cryptographic primitives used in PQKryvos, namely the BDLOP commitments from Section 4, can be expressed efficiently (i.e., with as few constraints as possible) as R1CS for the Ligero zkSNARK described in Section 3. To initialize PQKryvos efficiently, we can use the modular subcircuit structure already included in Kryvos (cf. Section 2) to treat the subcircuits used for evaluating choice space membership and for computing f_{res}, independently of the employed commitment scheme. The circuits for choice space membership and f_{res} can be directly used for Ligero. We therefore focus on the more complex sub-circuit for computing a commitment.

Encoding the Commitment Key as Part of the R1CS. Recall from (1) that a commitment is computed as $Ar + (0, x)^T$ for the public $(n \times t)$ matrix A. In particular, the computation of a commitment consists entirely of linear operations with publicly known scalars. We can therefore hard-code A into the R1CS matrices (as *constant values*) instead of treating it as part of the assignment vector (as *input values*), leveraging the fact that additions (and more generally linear operations) come, essentially, for free in an R1CS representation (as already explicitly stated in [47]).

More concretely, given a matrix $A \in \mathbb{F}^{t \times n}$, the relation $A \cdot r = y$, with $r \in \mathbb{F}^n$ being part of the assignment vector $\boldsymbol{z}$ of the R1CS, translates to t constraints of the form $\sum_{i=1}^{n} a_{ji} r_i = y_j$ for $j = 1, \ldots, t$ and a_{ij}, r_i, y_j the entries of A, r, y, respectively. Obviously, these constraints are linear in the variable, namely r, and do not leverage the quadratic structure of an R1CS equation at all. Hence, they can be merged with other constraints of r and do not affect the overall constraint count then.

In Kryvos, it seems that the commitment key, namely the analogue of A for (extended) Pedersen commitments, is encoded as input values instead of constants. In our setup, this would lead to one additional (non-linear) individual constraint for *each* of the n summands above for each commitment. Choosing A as public constants is therefore key to generating a small R1CS for expressing the computation of BDLOP commitments.

We remark that treating A as a variable in principle allows reusing the R1CS with different commitment keys. However, since in an execution of Kryvos (and PQKryvos) the commitment key is always fixed at the beginning, generating a corresponding R1CS can be done in advance, and repeated, for each election, or, more precisely, each new commitment key. Hence, the design choice used in Kryvos [16] was not optimal. We expect that Kryvos with Pedersen commitments could be improved by treating commitment keys as the constants they are; however, the gains are likely less significant than for the linear BDLOPs commitments in PQKryvos.

Avoiding Foreign Field Arithmetic. In general, expressing the arithmetic in $\mathbb{F}_q$ naturally used for BDLOP commitment in a different field $\mathbb{F}$ used by zkSNARKs introduces an overhead, as, in principle, one has to apply a reduction modulo q after each computation step. Each modulo reduction then leads to additional constraints. When $\mathbb{F}$ is sufficiently large, however, one can severely reduce the resulting overhead by only applying the modular reduction after several (ideally all) computation steps. Concretely, for computing a BDLOP commitment without intermediate modular reduction, one needs to assume that an entry of the commitment as defined by (1) does not exceed the modulus of $\mathbb{F}$. This is the case if $B_q = dNq^2 + 2q < |\mathbb{F}|$, where we used the specific form of the parameter matrix A in Section 4. For example, using the parameters proposed by del Pino et al. [72] for the instantiation of BDLOP commitments, i.e. $q \approx 2^{31}$, $N = 256$, and $d = 7$, a SNARK-field $\mathbb{F}$ of size $\geq 2^{73}$ is sufficient to postpone the modular reduction. Notably, this includes the scalar field $\mathbb{F}_{p_{\mathsf{BN254}}}$ of the pairing-friendly BN254, which was used for instantiating Groth16 in the original Kryvos instantiation. Hence, the combination of del Pino parameters and $\mathbb{F}_{p_{\mathsf{BN254}}}$ already avoids some of the costly pitfalls of working over two different fields.

However, by setting $\mathbb{F} = \mathbb{F}_q$, one can achieve much lower constraint numbers, since the overhead of expressing modular operations and enforcing size conditions on input values inside the R1CS is avoided completely. Recall from Section 3, that Ligero is not bound to a particular field. Hence, we could instantiate our system over the field $\mathbb{F}_q$ proposed by del Pino et al. [72]. However, as already stated in Section 3, we are practically restricted to fields that are supported by common circuit compilers, such as `Circom`. We hence consider the Goldilocks prime $p_{\mathsf{gl}} = 2^{64} - 2^{32} + 1$ and instantiate the BDLOP commitments over $\mathbb{F}_q = \mathbb{F}_{p_{\mathsf{gl}}}$ as described in Section 4.

The Overall Constraint System. Combining the above considerations, we are almost prepared to compute the exact overall number of constraints. We need, however, to still address the following two conditions: *(i)* The computation $A \cdot r + (0, x)^T$ is done modulo Φ_m and *(ii)* $||r||_\infty \leq \beta$. The first condition does not impose any further constraints. In fact, one can simply compute the polynomial products without modular reduction and then shift the high-degree coefficients according to the modulus Φ_m, which only consists of further additions that can be encoded in other, already specified constraints. In particular, the choice of m does not affect the number of constraints. We therefore chose m a power-of-three, which is slightly less conventional than powers-of-two, but supports both constructions outlined in Section 4.

For the second condition, for small values of $\beta > 0$, one can simply require $r_i \cdot \prod_{k=1}^{\beta}(r_i + q - k)(r_i - k) = 0$, which yields 2β constraints per coefficient, i.e., $2\beta \cdot (2d+1) \cdot N$ constraints in total.[12] When $\mathbb{F} = \mathbb{F}_q$, these are the only non-linear constraints. In the easiest case, where $\beta = 1$ and $\mathbb{F} = \mathbb{F}_q$, we hence need $2(2d+1) \cdot N$ overall constraints. As derived in Section 4, we can securely instantiate BDLOP commitments over $\mathbb{F}_{p_{\mathsf{gl}}}$ by setting $N = 486 = \varphi(3^6)$ and $d = 6$, yielding 13 000 constraints for the computation of a commitment to N values. Meanwhile, when using del Pino's parameters within $\mathbb{F}_{p_{\mathsf{BN254}}}$, the overhead caused by foreign field arithmetic is severe, and committing to $N = 256$ values already requires about 300 000 constraints. As a comparison, Huber et al. reported about 200 000 constraints for committing to 250 values with their instantiation of Pedersen commitments over $\mathbb{F}_{p_{\mathsf{BN254}}}$. We report the prover efficiency of the different combinations in Table 1.

Finally, with these parameters, our e-voting protocol achieves pq-security of 150 bits in time and 100 bits in space according to the current version of [73].

Comparison to Previous Approaches. Our new combination of BDLOP and Ligero directly yields NIZKPoKs of a valid opening. Previous approaches like [35] instead use highly specialized ZKPs of knowledge of an opening. While these ZKPs might sometimes be even smaller and faster to compute, our circuit-based approach offers far enhanced flexibility, which we exploit for PQKryvos. Beyond the direct application in PQKryvos, this flexibility also allows us to prove in zero-knowledge linear and quadratic relations between committed values. For example, we can prove that two commitments $c_1 \leftarrow \mathsf{Com}(x_1, r_1)$ and $c_2 \leftarrow \mathsf{Com}(x_2, r_2)$ fulfill $ax_1 + bx_2 = 0$ for two fixed elements $a, b \in \mathcal{R}_q$ by combining two copies of our commitment circuit and a subcomponent enforcing the linear relation. The latter can be easily realized in our implementation [45] since the arithmetic of the BDLOP message space is already available. We included this circuit as an illustrative example in [45]. Other specialized ZKPs, e.g., for products of messages as in [74], can be realized analogously with [45].

[12] For higher values of β there are more efficient ways of ensuring size conditions

Table 1: Comparison of constraint numbers and proving times for different combinations of commitments (to N values), primes, and GPZKPs. The benchmarks in [16] were obtained using a machine comparable to ours.[13]

Construction	Constraints	Prove[s]
BDLOP + p_{gl} (Ligero, $N = 486$)	$\sim 13\,000$	~ 1.5s
BDLOP + p_{BN254} (Ligero, $N = 486$)	$\sim 300\,000$	~ 18s
Pedersen + p_{BN254} [16] (Groth16,$N = 250$)	$\sim 200\,000$	22.4s
Pedersen + p_{BN254} [16] (Groth16,$N = 10$)	$\sim 14\,500$	~ 1.7s

5.3 Benchmarks for Various Choice Spaces

Based on the observations above, we have implemented and evaluated the Ligero proofs used in PQKryvos for evaluating π_{ballot} and π_{result} with BDLOP commitments over $\mathbb{F}_{p_{\mathsf{gl}}}$ for various choice spaces and result functions, which we describe in detail in Appendix B. For this purpose, we have used `circom` [46] to design arithmetic circuits for computing BDLOP commitments and leverage the modularity of the original Kryvos circuits to build full circuits for computing f_{res} and for ensuring ballot validity. Based on the resulting R1CSs, we use the `libiop` implementation of Ligero [59] to benchmark proving/verification times, and proof sizes.

We note that, since in the Goldilocks case we use $N = 486$, a single commitment can accommodate almost all realistic candidate numbers. Even in the case of Condorcet methods, where a ballot is a $n_{\mathsf{cand}} \times n_{\mathsf{cand}}$ matrix, up to $n_{\mathsf{cand}} = 22$ candidates are covered with a single commitment – much more than one would usually have in such elections.

For higher numbers, additional commitment subcircuits are required, and the relations $\mathfrak{R}_{\mathsf{ballot}}$ and $\mathfrak{R}_{f_{\mathsf{res}}}$ need to be adapted accordingly (see [16] for a similar discussion). Given the small size of the commitment subcircuit ($\sim 13\,000$ constraints), this remains practical for moderate numbers of commitments. A notable application is the case of IRV. Here, following Kryvos, we encode an IRV ballot as a single-choice vote for a (potentially partial) ranking of the n_{cand} candidates, yielding more than $n_{\mathsf{choices}} = n_{\mathsf{cand}}!$ choices. For this rapidly growing choice space, as soon as more than 5 candidates are to be ranked, one commitment does not suffice for encoding a vote. We therefore have evaluated the performance of Ligero for up to 28 commitments, which can fit 14 000 values and, hence, encode partial rankings of up to 7 candidates. The associated circuit for proving $\mathfrak{R}_{\mathsf{ballot}}$ still requires only about 370 000 constraints and the corresponding Ligero proof π_{ballot} was created on a standard laptop[13] in 35 seconds. The largest number of IRV candidates supported by the original Kryvos instantiation is 6, which already required splitting the proof into ten substatements and treating them separately with multiple Groth16 provers running in parallel. The latter technique is also possible in PQKryvos but requires more computational resources of the voters.

We provide benchmarks for computing π_{ballot} and π_{result} with Ligero and BDLOP commitments for all considered choice spaces and result functions (except for IRV, which we discussed above) in Fig. 1. Our benchmarks show that PQKryvos is highly efficient even for large numbers of candidates and complex choice spaces and result functions, with proof computation never requiring more than 30 s (and rarely exceeding 10 s) for realistic candidate numbers.

In Fig. 2 we compare some of our benchmarks with the corresponding ones of the original Kryvos instantiation from [16, 40].

[13] We have used a Lenovo ThinkPad with an Intel i7-6600U CPU running at 2.60GHz and 12GB of RAM running Ubuntu 22.04 LTS for all our benchmarks.

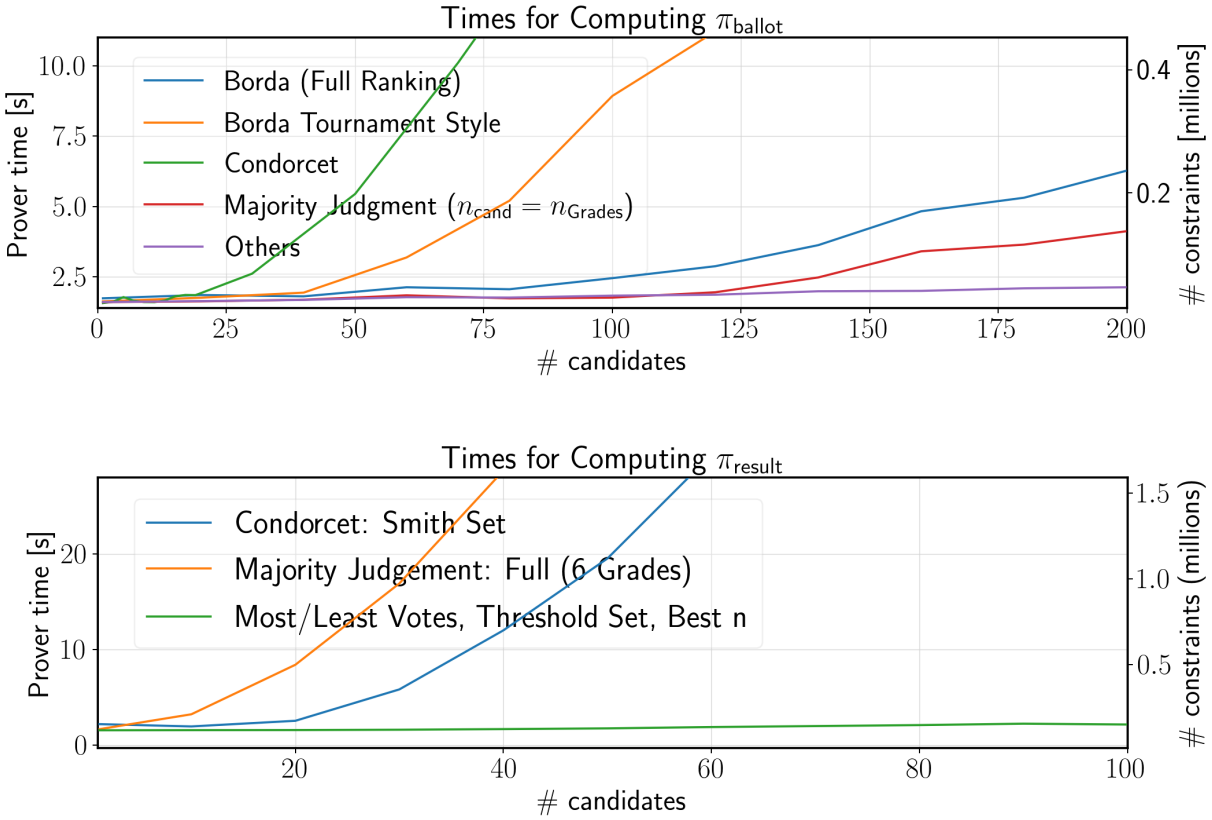

Fig. 1: Prover runtime and constraint numbers for computing π_{ballot} (top) and π_{result} (bottom) for several voting methods/result functions. Others include Majority Judgment for 6 grades, Borda with 10 ranks, Multi-Vote and Single-Vote. See Appendix B for details.

In all cases, our computations of both, π_{ballot} and π_{result}, are much faster than the ones from the original Kryvos instantiation. For example, computing π_{ballot} for a Condorcet ballot for 20 candidates we required 1.6 seconds, where Kryvos reports about 30 seconds. Similar benchmarks apply for the computation of a π_{result} for proving the correct computation of a Condorcet Smith set for 20 candidates. As another example, the computation of π_{result} for IRV with 6 candidates takes us less than 10 seconds with PQKryvos, where Kryvos reports about 3 minutes. Since we leverage the same sub-circuits as Kryvos, this performance improvement can be mostly attributed to the smaller R1CS possible with BDLOP commitments, cf. Table 1. We note that a well-known disadvantage of Ligero compared to Groth16 is its generally larger proof size and (in turn) also larger verification time. For PQKryvos this means that we need proof sizes of several kilobytes (up to 1 MB depending on the circuit) and verification times of up to 30 seconds, while Groth16 has proof sizes of $<$ 1KB and verification times of less than a second. We think that these disadvantages are often not relevant for e-voting, where the verification of π_{ballot} can take place any time after a ballot is cast. While we favor Ligero over Groth16 as explained in Section 3 (particularly, because it is pq-secure), the Kryvos framework could also be initiated with Groth16 and BDLOP commitments over the scalar field of BN254, which then leads to correspondingly lower proof sizes and verification times. We have seen in Section 4 that the underlying theory still works then. Unfortunately, the generation of R1CS for BN254 with `circom` was not possible on our standard laptop due to memory issues, and we therefore decided to provide benchmarks only for p_{gl}. However, the final R1CS for BN254 will be similarly small as the one we used for p_{gl}.

Comparison to Existing PQ-Secure Protocols. Finally, we compare our results to the established homomorphic pq-secure e-voting systems mentioned in Section 1. The EVOLVE system by del Pino et al. [32] and its verifiable extension Epoque [33] report 8.5 ms for computing Ballot validity proofs for the case of YES/NO Voting with 1 candidate/voting option on a machine comparable to the one we used for our benchmarks. The reported proof sizes are roughly 15 KB per tallier, which becomes around 60 KB for typically 4 talliers in [32]. In PQKryvos we need about 1.5 s for the same task – a difference likely not noticed by voters when casting their ballot. Both EVOLVE and Epoque are limited to YES/NO elections with 1 candidate.[14] Other approaches like Farzaliyev et al. [36] present protocols specifically tailored for Multi-Vote. The paper indicates that for 2 candidates, a proof of ballot validity requires about 0.17 s. It is unclear what exactly is benchmarked and how this performance scales for the higher candidate numbers we support. As mentioned before, none of the currently available pq-secure protocols support tally-hiding. However, one could use our GPZKPs to directly extend EVOLVE and Epoque in order to support public tally-hiding by applying f_{res} to the election result and computing π_{result} just as we do. Moreover, as already explained in Section 4, using GPZKPs for ballot validity makes PQKryvos no longer require the ZKPs for randomness reduction that EVOLVE and Epoque use. While from the description in [32] it is unclear how these ZKPs affect the efficiency of the tallying phase in general; examples suggest that our tallying phase is significantly faster. For example, Del Pino et al. [32] provide benchmarks for tallying an election with 11 000 voters, which requires 41 min for the talliers' ZKPs with an amortization over 4000 votes, i.e. we would save 41 min in this example and likely more in elections with more voters.

6 Security Analysis of PQKryvos

We formally analyze the security of PQKryvos within the established KTV framework for verifiability and privacy [12, 14, 75] and its extension to public tally-hiding [16]. We briefly recall the definitions and proof sketches here, while providing the detailed security analysis of PQKryvos and all formal background in Appendix C. Overall, we show the following theorem, which is a combination of Theorem 5 and Theorem 6 from below and the security of the involved cryptographic primitives:

Theorem 4 (Security of PQKryvos). *The PQKryvos protocol with pq-secure instantiations of Ligero, BDLOP, and the IND-CCA2-secure encryption scheme* E *is a post-quantum secure verifiable and publicly tally-hiding e-voting protocol with respect to* f_{res}.

[14] The papers describe an extension to YES/NO elections with more than one candidate, but provide no performance analysis.

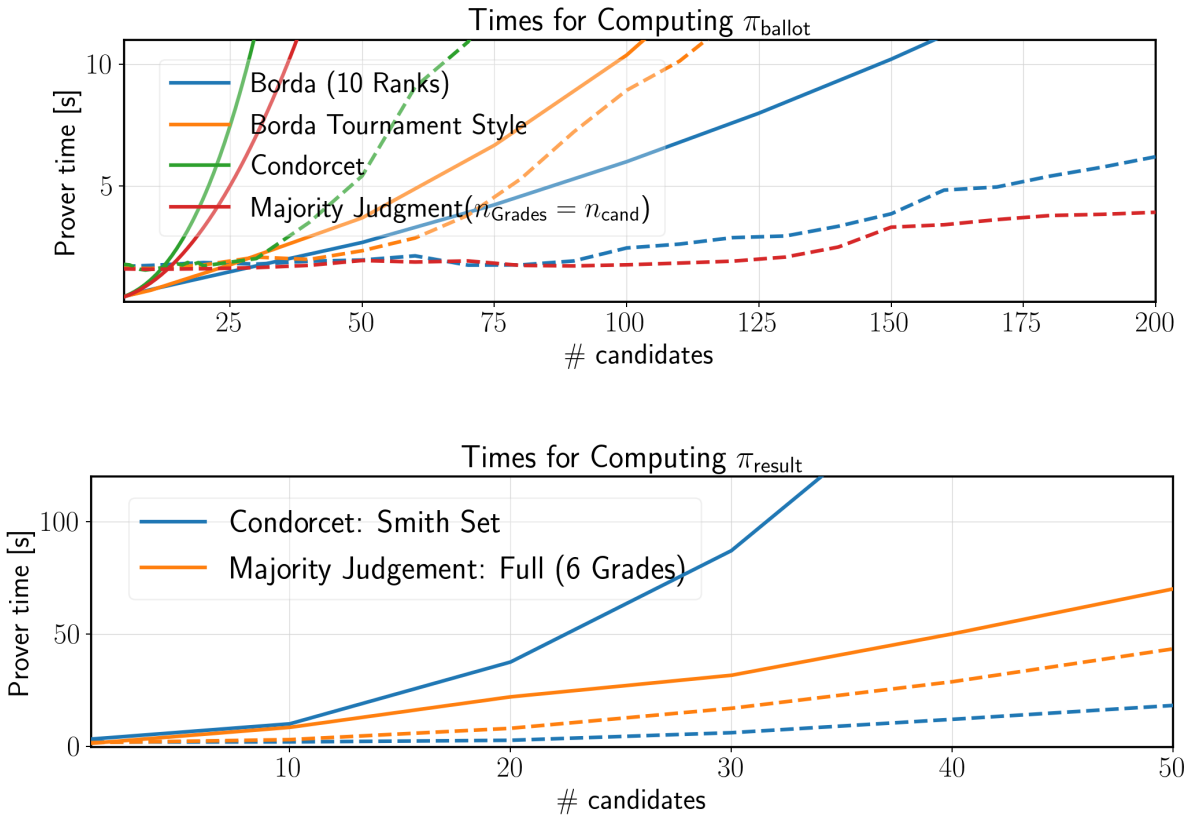

Fig. 2: Comparison of prover runtimes for compuing π_{ballot} (top) and π_{result} (bottom) for various choice spaces/result functions between Kryvos (solid) and PQKryvos (dashed).

Our security analysis mostly follows the same steps as the one for Kryvos [16], with the following differences: *(i)* Our analysis always considers ppt quantum adversaries, *(ii)* our analysis also accounts for the (negligible) decryption error that many pq-secure encryption schemes introduce, and *(iii)* our privacy proof does neither require perfect zero-knowledge nor perfectly hiding commitments, both of which were stated as requirements for the privacy analysis in [16].

6.1 Verifiability

Verifiability Definition. In the KTV framework, verifiability is defined with respect to a formally specified goal γ, which is a set of protocol runs. A virtual honest entity called J (the *judge*) performs protocol-specific checks and outputs either `accept` or `reject`. J takes as input only public information, modelling that any party (e.g., voters or external observers) could perform the judge's checks. A goal is considered verifiable if, for every (ppt quantum) adversary, the probability that a protocol run violates the goal while the judge nevertheless outputs `accept` is negligible in the security parameter.

Verifiability of PQKryvos. To analyze the verifiability of PQKryvos, we assume a further protocol participant S (the *scheduler*) that subsumes the voting authority Auth and schedules the protocol phases. We consider the goal $\gamma(\varphi)$ (formally defined in Appendix C) that includes all runs of PQKryvos where, if BB and S are honest, a correct election result (i.e., a result that is consistent with the honest voters' actual votes and valid votes of dishonest voters) is returned. We give a full proof of the following theorem in Appendix C.

Theorem 5 (Verifiability of PQKryvos). *If the encryption scheme* E *is correct up to a negligible decryption error, the commitment scheme* C *is additively homomorphic and computationally binding against ppt quantum adversaries, the GPZKP* Π *is complete and computationally sound against ppt quantum adversaries, and the NIZKP* π_{dec} *is computationally sound against ppt quantum adversaries, then the goal* $\gamma(\varphi)$ *is verifiable in* $\mathsf{P}_{\mathsf{PQKryvos}}$ *by the judge* J *that reads all data from* BB *and writes* `accept` *in a run if and only if* (i) *all NIZKPs* π_{dec} *of correct decryption on* BB *are valid (if existent), and* (ii) *the NIZKPoK* π_{result} *of correct result computation corresponding to the aggregated commitment* c *obtained by adding all voters' commitments on* BB *for which no tallier complaint alongside a valid* π_{dec} *has been sent.*

Proof Sketch. The core arguments of the verifiability analysis are as follows: Since BB is honest, honest votes are preserved, and invalid votes are discarded – except with negligible probability due to the soundness error of Π. The probability that an honest voter's ciphertext does not decrypt to a valid commitment opening is bounded by the encryption error of E and, hence, negligible. Any incorrect decryption or falsely claimed result by dishonest talliers is detectable except with negligible probability due to the soundness errors of Π, and π_{dec}. Hence, J accepts only results consistent with honest votes and valid dishonest votes, up to negligible error.

6.2 Privacy and Public Tally-Hiding

Definition of Privacy and Public Tally-Hiding. We analyze the privacy of PQKryvos in the KTV framework using a game-based variant of their definition [16,17,76]. Privacy is defined via an indistinguishability game in which an adversary must distinguish whether a given voter $\mathsf{V}_{\mathsf{obs}}$ (the *voter under observation*) cast one of two chosen votes. Since an election result may always leak some information, privacy is defined relative to an ideal protocol $\mathsf{P}^{\mathsf{ideal}}$ that reveals only the election result. We say that a protocol achieves *ideal privacy* if, for every considered adversary in the real protocol, there exists a corresponding adversary in the ideal setting such that the difference in success probabilities is negligible.

We additionally evaluate public tally-hiding in the KTV framework, which is formalized via two separate privacy requirements: *Public privacy* requires that if all talliers are honest, nothing beyond the final result is revealed. *Internal privacy* requires that if at least one tallier is honest, nothing beyond the aggregated tally is revealed. Both are formalized using the privacy definition explained above, but differ in the ideal protocol and the considered adversaries. We refer to Appendix C.3 for details.

Privacy and Public Tally-Hiding of PQKryvos. We give a detailed proof of the following theorem in Appendix C.3. Here, A_{int} denotes the set of adversaries considered for internal privacy and A_{pub} denotes the set of adversaries considered for public privacy.

Theorem 6 (Privacy of PQKryvos). *Let A_{int} consist of all (ppt quantum) adversaries that do not corrupt* BB *or* S*, corrupt at most $n^{\mathsf{hon}}_{\mathsf{voters}}$ voters, and corrupt at most all but one tallier. Let $A_{\mathsf{pub}} = \{\mathcal{A} \in A_{\mathsf{int}} \mid \mathcal{A}$ corrupts no tallier$\}$.*

If the encryption scheme E *is IND-CCA2-secure and correct up to a negligible decryption error, the GPZKP Π achieves computational zero-knowledge, the NIZKP π_{dec} achieves computational zero-knowledge, and the commitment scheme* C *is computational hiding (all against ppt quantum adversaries), then* PQKryvos *achieves public tally-hiding w.r.t. $A_{\mathsf{int}}, A_{\mathsf{pub}}$.*

In particular, in the non-tally-hiding case ($f_{\mathsf{res}} = \mathsf{id}$), PQKryvos *achieves ideal privacy w.r.t. adversaries in A_{int} (cf. Remark 5).*

Proof Sketch. We prove both properties via a sequence of games in which PQKryvos is replaced by an ideal protocol. By the zero-knowledge property of Π and π_{dec}, honest voters' and talliers' ZKPs can be replaced with simulated proofs with only a negligible effect on an adversary's winning probability. By IND-CCA2 security of E and hiding of C, honest voters can replace ciphertexts and commitments with dummy values while forwarding the actual values internally to honest talliers, again affecting the winning probability negligibly. At this point, all adversary-observable information, except the election result (public privacy) or aggregated tally (internal privacy), is independent of honest votes, allowing replacement of PQKryvos with the ideal protocol $\mathsf{P}^{\mathsf{ideal}}$ with negligible difference in the adversary's winning probability.

7 Conclusion

In this work, we proposed PQKryvos, the first post-quantum secure e-voting protocol that achieves public tally-hiding and supports flexible ballot formats. We integrated BDLOP commitments with the Ligero GPZKP, and therewith construct efficient pq-secure ZKPs for ballot validity and result computation. This approach enables publicly tally-hiding elections for virtually any voting method, including Single- and Multi-Vote, Borda count, Majority Judgment, and IRV. Our design resolves key compatibility challenges, demonstrating that BDLOP commitments are efficiently combinable with R1CS-based zkSNARKs and can be instantiated over SNARK-friendly primes, such as Goldilocks. Moreover, our results show that using Ligero, one can avoid the computational overhead caused by expensive verifiable re-randomization of BDLOP commitments, which significantly reduces the cost of the tallying phase over existing pq-secure e-voting protocols that use BDLOP commitments [32, 33].

Our implementation and benchmarks show that PQKryvos is practical for real-world elections and even outperforms the original, pre-quantum instantiation of Kryvos in most cases. As a result, PQKryvos closes an important gap between the theory and practice of post-quantum secure e-voting, provides an efficient, flexible foundation for future remote election systems, and ensures privacy and verifiability even in the presence of quantum adversaries.

Acknowledgements:

This research was supported by the German Federal Ministry of Education and Research (QCyber under grant agreement 16KIS2590K), and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grants 411720488 and 548713845. The authors thank Felix Röhr for support with the implementation.

References

1. Ben Adida, Olivier de Marneffe, Olivier Pereira, and Jean-Jaques Quisquater. Electing a University President Using Open-Audit Voting: Analysis of Real-World Use of Helios. In *USENIX/ACCURATE Electronic Voting Technology (EVT 2009)*, 2009.

2. Johannes Müller. Breaking and fixing vote privacy of the estonian e-voting protocol IVXV. In Shin'ichiro Matsuo, Lewis Gudgeon, Ariah Klages-Mundt, Daniel Perez Hernandez, Sam Werner, Thomas Haines, Aleksander Essex, Andrea Bracciali, and Massimiliano Sala, editors, *Financial Cryptography and Data Security. FC 2022 International Workshops - CoDecFin, DeFi, Voting, WTSC, Grenada, May 6, 2022, Revised Selected Papers*, volume 13412 of *Lecture Notes in Computer Science*, pages 325–334. Springer, 2022.
3. Kristian Gjøsteen. The norwegian internet voting protocol. In Aggelos Kiayias and Helger Lipmaa, editors, *E-Voting and Identity - Third International Conference, VoteID 2011, Tallinn, Estonia, September 28-30, 2011, Revised Selected Papers*, volume 7187 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2011.
4. Ella Kummer, Bryan Ford, and Louis-Henri Merino. Swiss post e-voting. 2022.
5. NSW Government. Constitution Act No 32. `https://legislation.nsw.gov.au/~/view/act/1902/32`, 2020.
6. Jack Santucci. Maine ranked-choice voting as a case of electoral-system change. *Representation*, 54(3):297–311, 2018.
7. Alexandre Debant and Lucca Hirschi. Reversing, breaking, and fixing the french legislative election e-voting protocol. In Joseph A. Calandrino and Carmela Troncoso, editors, *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, pages 6737–6752. USENIX Association, 2023.
8. Thomas Haines, Sarah Jamie Lewis, Olivier Pereira, and Vanessa Teague. How not to prove your election outcome. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 644–660. IEEE, 2020.
9. Rolf Haenni. Swiss post public intrusion test: Undetectable attack against vote integrity and secrecy. *Bern Univ. Appl. Sci. Biel, Switzer land*, 2019.
10. J. Alex Halderman and Vanessa Teague. The new south wales ivote system: Security failures and verification flaws in a live online election. In Rolf Haenni, Reto E. Koenig, and Douglas Wikström, editors, *E-Voting and Identity - 5th International Conference, VoteID 2015, Bern, Switzerland, September 2-4, 2015, Proceedings*, volume 9269 of *Lecture Notes in Computer Science*, pages 35–53. Springer, 2015.
11. Drew Springall, Travis Finkenauer, Zakir Durumeric, Jason Kitcat, Harri Hursti, Margaret MacAlpine, and J. Alex Halderman. Security analysis of the estonian internet voting system. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 703–715. ACM, 2014.
12. Véronique Cortier, David Galindo, Ralf Küsters, Johannes Müller, and Tomasz Truderung. Sok: Verifiability notions for e-voting protocols. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 779–798. IEEE Computer Society, 2016.
13. Der Bundesrat. Sicherheit beim E-Voting. `https://www.bk.admin.ch/bk/de/home/politische-rechte/e-voting/sicherheit-beim-e-voting.html`, 2018.
14. Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Verifiability, privacy, and coercion-resistance: New insights from a case study. In *32nd IEEE Symposium on Security and Privacy, SP 2011, 22-25 May 2011, Berkeley, California, USA*, pages 538–553. IEEE Computer Society, 2011.
15. David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi. Sok: A comprehensive analysis of game-based ballot privacy definitions. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 499–516. IEEE Computer Society, 2015.
16. Nicolas Huber, Ralf Küsters, Toomas Krips, Julian Liedtke, Johannes Müller, Daniel Rausch, Pascal Reisert, and Andreas Vogt. Kryvos: Publicly tally-hiding verifiable e-voting. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 1443–1457. ACM, 2022.
17. Ralf Küsters, Julian Liedtke, Johannes Müller, Daniel Rausch, and Andreas Vogt. Ordinos: A verifiable tally-hiding e-voting system. In *IEEE European Symposium on Security and Privacy, EuroS&P 2020, Genoa, Italy, September 7-11, 2020*, pages 216–235. IEEE, 2020.
18. Alan Szepieniec and Bart Preneel. New techniques for electronic voting. In *USENIX Journal of Election Technology and Systems (JETS)*, 2015.
19. Véronique Cortier, Pierrick Gaudry, and Quentin Yang. A toolbox for verifiable tally-hiding e-voting systems. In Vijayalakshmi Atluri, Roberto Di Pietro, Christian Damsgaard Jensen, and Weizhi Meng, editors, *Computer Security - ESORICS 2022 - 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26-30, 2022, Proceedings, Part II*, volume 13555 of *Lecture Notes in Computer Science*, pages 631–652. Springer, 2022.
20. Gorjan Alagic, Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl Miller, et al. Status report on the third round of the nist post-quantum cryptography standardization process. 2022.

21. Valeh Farzaliyev, Jan Willemson, and Jaan Kristjan Kaasik. Improved lattice-based mix-nets for electronic voting. *IET Inf. Secur.*, 17(1):18–34, 2023.
22. Diego F. Aranha, Carsten Baum, Kristian Gjøsteen, and Tjerand Silde. Verifiable mix-nets and distributed decryption for voting from lattice-based assumptions. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, pages 1467–1481. ACM, 2023.
23. Xavier Boyen, Thomas Haines, and Johannes Müller. A verifiable and practical lattice-based decryption mix net with external auditing. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *Computer Security - ESORICS 2020 - 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14-18, 2020, Proceedings, Part II*, volume 12309 of *Lecture Notes in Computer Science*, pages 336–356. Springer, 2020.
24. Diego F. Aranha, Carsten Baum, Kristian Gjøsteen, Tjerand Silde, and Thor Tunge. Lattice-based proof of shuffle and applications to electronic voting. In Kenneth G. Paterson, editor, *Topics in Cryptology - CT-RSA 2021 - Cryptographers' Track at the RSA Conference 2021, Virtual Event, May 17-20, 2021, Proceedings*, volume 12704 of *Lecture Notes in Computer Science*, pages 227–251. Springer, 2021.
25. Javier Herranz, Ramiro Martínez, and Manuel Sánchez. Shorter lattice-based zero-knowledge proofs for the correctness of a shuffle. In Matthew Bernhard, Andrea Bracciali, Lewis Gudgeon, Thomas Haines, Ariah Klages-Mundt, Shin'ichiro Matsuo, Daniel Perez, Massimiliano Sala, and Sam Werner, editors, *Financial Cryptography and Data Security. FC 2021 International Workshops - CoDecFin, DeFi, VOTING, and WTSC, Virtual Event, March 5, 2021, Revised Selected Papers*, volume 12676 of *Lecture Notes in Computer Science*, pages 315–329. Springer, 2021.
26. Núria Costa, Ramiro Martínez, and Paz Morillo. Lattice-based proof of a shuffle. In Andrea Bracciali, Jeremy Clark, Federico Pintore, Peter B. Rønne, and Massimiliano Sala, editors, *Financial Cryptography and Data Security - FC 2019 International Workshops, VOTING and WTSC, St. Kitts, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers*, volume 11599 of *Lecture Notes in Computer Science*, pages 330–346. Springer, 2019.
27. Núria Costa, Ramiro Martínez, and Paz Morillo. Proof of a shuffle for lattice-based cryptography. In Helger Lipmaa, Aikaterini Mitrokotsa, and Raimundas Matulevicius, editors, *Secure IT Systems - 22nd Nordic Conference, NordSec 2017, Tartu, Estonia, November 8-10, 2017, Proceedings*, volume 10674 of *Lecture Notes in Computer Science*, pages 280–296. Springer, 2017.
28. Patrick Hough, Caroline Sandsbråten, and Tjerand Silde. More efficient lattice-based electronic voting from NTRU. *IACR Commun. Cryptol.*, 1(4):10, 2024.
29. Thomas Haines, Rafieh Mosaheb, Johannes Müller, and Ivan Pryvalov. Sok: Secure e-voting with everlasting privacy. *Proc. Priv. Enhancing Technol.*, 2023(1):279–293, 2023.
30. Edouard Cuvelier, Olivier Pereira, and Thomas Peters. Election verifiability or ballot privacy: Do we need to choose? In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings*, volume 8134 of *Lecture Notes in Computer Science*, pages 481–498. Springer, 2013.
31. Véronique Cortier, Pierrick Gaudry, and Stéphane Glondu. Belenios: A Simple Private and Verifiable Electronic Voting System. In *Foundations of Security, Protocols, and Equational Reasoning - Essays Dedicated to Catherine A. Meadows*, volume 11565 of *Lecture Notes in Computer Science*, pages 214–238. Springer, 2019.
32. Rafaël del Pino, Vadim Lyubashevsky, Gregory Neven, and Gregor Seiler. Practical Quantum-Safe Voting from Lattices. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1565–1581. ACM, 2017.
33. Xavier Boyen, Thomas Haines, and Johannes Müller. Epoque: Practical End-to-End Verifiable Post-Quantum-Secure E-Voting. In *IEEE European Symposium on Security and Privacy, EuroS&P 2021, Vienna, Austria, September 6-10, 2021*, pages 272–291. IEEE, 2021.
34. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. A homomorphic LWE based e-voting scheme. In Tsuyoshi Takagi, editor, *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings*, volume 9606 of *Lecture Notes in Computer Science*, pages 245–265. Springer, 2016.
35. Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More efficient commitments from structured lattice assumptions. In Dario Catalano and Roberto De Prisco, editors, *Security and Cryptography for Networks - 11th International Conference, SCN 2018, Amalfi, Italy, September 5-7, 2018, Proceedings*, volume 11035 of *Lecture Notes in Computer Science*, pages 368–385. Springer, 2018.
36. Valeh Farzaliyev, Calvin Pärn, Heleen Saarse, and Jan Willemson. Lattice-based zero-knowledge proofs in action: Applications to electronic voting. *J. Cryptol.*, 38(1):6, 2025.

37. Miranda Christ, Foteini Baldimtsi, Konstantinos Kryptos Chalkias, Deepak Maram, Arnab Roy, and Joy Wang. Sok: Zero-knowledge range proofs. In Rainer Böhme and Lucianna Kiffer, editors, *6th Conference on Advances in Financial Technologies, AFT 2024, September 23-25, 2024, Vienna, Austria*, volume 316 of *LIPIcs*, pages 14:1–14:25. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.
38. Jens Groth. Non-interactive Zero-Knowledge Arguments for Voting. In *Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings*, volume 3531 of *Lecture Notes in Computer Science*, pages 467–482, 2005.
39. Rui Joaquim. How to prove the validity of a complex ballot encryption to the voter and the public. *J. Inf. Secur. Appl.*, 19(2):130–142, 2014.
40. Nicolas Huber, Ralf Küsters, Julian Liedtke, and Daniel Rausch. Zk-snarks for ballot validity: A feasibility study. In David Duenas-Cid, Peter B. Roenne, Melanie Volkamer, Jurlind Budurushi, Michelle L. Blom, Adrià Rodríguez-Pérez, Iuliia Spycher-Krivonosova, Jordi Castellà-Roca, and Jordi Barrat Esteve, editors, *Electronic Voting - 9th International Joint Conference, E-Vote-ID 2024, Tarragona, Spain, October 2-4, 2024, Proceedings*, volume 15014 of *Lecture Notes in Computer Science*, pages 107–123. Springer, 2024.
41. Felix Röhr, Nicolas Huber, and Ralf Küsters. Improving the efficiency of zkSNARKs for ballot validity. Cryptology ePrint Archive, Paper 2025/2286, 2025.
42. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, page 309–325, New York, NY, USA, 2012. Association for Computing Machinery.
43. National Institute of Standards and Technology. Module-lattice-based key-encapsulation mechanism standard, 2025.
44. Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight Sublinear Arguments Without a Trusted Setup. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 2087–2104. ACM, 2017.
45. Nicolas Huber, Pascal Reisert, and Ralf Küsters. Implementation of pqkryvos, 2025. Available at `https://github.com/HicolasNuber/pqkryvos`.
46. iden3. Circom. `https://github.com/iden3/circom`, 2025.
47. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 305–326. Springer, 2016.
48. Anca Nitulescu. Lattice-based zero-knowledge snargs for arithmetic circuits. In Peter Schwabe and Nicolas Thériault, editors, *Progress in Cryptology - LATINCRYPT 2019 - 6th International Conference on Cryptology and Information Security in Latin America, Santiago de Chile, Chile, October 2-4, 2019, Proceedings*, volume 11774 of *Lecture Notes in Computer Science*, pages 217–236. Springer, 2019.
49. Yuval Ishai, Hang Su, and David J. Wu. Shorter and faster post-quantum designated-verifier zksnarks from lattices. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 212–234. ACM, 2021.
50. Ward Beullens and Gregor Seiler. Labrador: Compact proofs for R1CS from module-sis. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part V*, volume 14085 of *Lecture Notes in Computer Science*, pages 518–548. Springer, 2023.
51. Dan Boneh and Binyi Chen. Latticefold: A lattice-based folding scheme and its applications to succinct proof systems. *IACR Cryptol. ePrint Arch.*, page 257, 2024.
52. Giacomo Fenzi, Christian Knabenhans, Ngoc Khanh Nguyen, and Duc Tu Pham. Lova: Lattice-based folding scheme from unstructured lattices. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology - ASIACRYPT 2024 - 30th International Conference on the Theory and Application of Cryptology and Information Security, Kolkata, India, December 9-13, 2024, Proceedings, Part IV*, volume 15487 of *Lecture Notes in Computer Science*, pages 303–326. Springer, 2024.
53. Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 103–128. Springer, 2019.

54. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable zero knowledge with no trusted setup. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 701–732. Springer, 2019.
55. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 14:1–14:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
56. Alexander Golovnev, Jonathan Lee, Srinath T. V. Setty, Justin Thaler, and Riad S. Wahby. Brakedown: Linear-time and field-agnostic snarks for R1CS. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part II*, volume 14082 of *Lecture Notes in Computer Science*, pages 193–226. Springer, 2023.
57. Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. Hyperplonk: Plonk with linear-time prover and high-degree custom gates. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part II*, volume 14005 of *Lecture Notes in Computer Science*, pages 499–530. Springer, 2023.
58. Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *IACR Cryptol. ePrint Arch.*, page 953, 2019.
59. scipr-lab. libiop. `https://github.com/scipr-lab/libiop`, 2021.
60. Ruihan Wang, Carmit Hazay, and Muthuramakrishnan Venkitasubramaniam. Ligetron: Lightweight scalable end-to-end zero-knowledge proofs post-quantum zk-snarks on a browser. In *IEEE Symposium on Security and Privacy, SP 2024, San Francisco, CA, USA, May 19-23, 2024*, pages 1760–1776. IEEE, 2024.
61. Inc. Ligero. Ligero prover, 2025. `https://github.com/ligeroinc/ligero-prover`, Ver.: 1.4.
62. Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 31–60, 2016.
63. Martijn Brehm, Binyi Chen, Ben Fisch, Nicolas Resch, Ron D. Rothblum, and Hadas Zeilberger. Blaze: Fast snarks from interleaved RAA codes. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part IV*, volume 15604 of *Lecture Notes in Computer Science*, pages 123–152. Springer, 2025.
64. Alessandro Chiesa, Peter Manohar, and Nicholas Spooner. Succinct arguments in the quantum random oracle model. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 1–29. Springer, 2019.
65. Lov K. Grover. A fast quantum mechanical algorithm for database search. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 212–219. ACM, 1996.
66. Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum algorithm for the collision problem. In *Encyclopedia of Algorithms*, pages 1662–1664. 2016.
67. Vadim Lyubashevsky and Gregor Seiler. Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 204–224, Cham, 2018. Springer International Publishing.
68. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Shorter lattice-based zero-knowledge proofs via one-time commitments. In Juan A. Garay, editor, *Public-Key Cryptography - PKC 2021 - 24th IACR International Conference on Practice and Theory of Public Key Cryptography, Virtual Event, May 10-13, 2021, Proceedings, Part I*, volume 12710 of *Lecture Notes in Computer Science*, pages 215–241. Springer, 2021.
69. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 553–572. Springer, 2010.
70. Raymond K Zhao, Sarah McCarthy, Ron Steinfeld, Amin Sakzad, and Máire O'Neill. Quantum-safe hibe: does it cost a latte? *IEEE Transactions on Information Forensics and Security*, 19:2680–2695, 2023.

71. Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over ntru lattices. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 22–41. Springer, 2014.
72. Ian Black, Emma McFall, Juliet Whidden, Bryant Xie, and Ryann Cartor. Practical quantum-safe voting from lattices, extended. *IACR Cryptol. ePrint Arch.*, page 1686, 2022.
73. Sam Scott Martin R. Albrecht, Rachel Player. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9:169–203, 2015. `https://github.com/malb/lattice-estimator`, Ver.: Sep 17, 2025, Commit: 352ddaf4a288a0543f5d9eb588d2f89c7acec463.
74. Thomas Attema, Vadim Lyubashevsky, and Gregor Seiler. Practical product proofs for lattice commitments. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, pages 470–499, Cham, 2020. Springer International Publishing.
75. Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Accountability: definition and relationship to verifiability. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 526–535. ACM, 2010.
76. Daniel Rausch, Nicolas Huber, and Ralf Küsters. Verifiable e-voting with a trustless bulletin board. In *38th IEEE Computer Security Foundations Symposium, CSF 2025, Santa Cruz, CA, USA, June 16-20, 2025*, pages 457–472. IEEE, 2025.
77. Fabrice Benhamouda, Jan Camenisch, Stephan Krenn, Vadim Lyubashevsky, and Gregory Neven. Better zero-knowledge proofs for lattice encryption and their application to group signatures. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, pages 551–572, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.
78. Rafael del Pino and Vadim Lyubashevsky. Amortization with fewer equations for proving knowledge of small secrets. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, pages 365–394, Cham, 2017. Springer International Publishing.
79. Ivan B. Damgård, Torben P. Pedersen, and Birgit Pfitzmann. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. In Douglas R. Stinson, editor, *Advances in Cryptology — CRYPTO' 93*, pages 250–265, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.
80. Republic of Nauru. Electoral Act 2016. `https://election.com.nr/wp-content/uploads/2022/08/Electoral-Act-2016-.pdf`, 2016.
81. Michel Balinski and Rida Laraki. A theory of measuring, electing, and ranking. *Proceedings of the National Academy of Sciences*, 104(21):8720–8725, 2007.
82. Aggelos Kiayias, Annabell Kuldmaa, Helger Lipmaa, Janno Siim, and Thomas Zacharias. On the security properties of e-voting bulletin boards. In Dario Catalano and Roberto De Prisco, editors, *Security and Cryptography for Networks - 11th International Conference, SCN 2018, Amalfi, Italy, September 5-7, 2018, Proceedings*, volume 11035 of *Lecture Notes in Computer Science*, pages 505–523. Springer, 2018.
83. Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Clash attacks on the verifiability of e-voting systems. In *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA*, pages 395–409. IEEE Computer Society, 2012.

A Proofs and Further Technical Results

Theorem 2. *The commitment scheme* $\mathsf{BDLOP}(q, N, d, \beta, B_r)$ *described above is computationally hiding if* $\mathsf{MLWE}(q, N, d, S_{\infty,\beta})$ *is computationally hard. It is computationally binding for the classical opening if* $\mathsf{MSIS}(q, N, d, 2d+1, 2B_r, A_1)$ *is computationally hard. If every element of* $\overline{\mathcal{C}}$ *is invertible, then it is computationally binding for the approximate opening if* $\mathsf{MSIS}(q, N, d, 2d+1, 4\sqrt{60}B_r, A_1)$ *is computationally hard.*

Proof. Consider an adversary $\mathcal{A}$ breaks the binding property, i.e. finds c, x, x', r, r' with valid messages $x \neq x'$ and randomness r, r' such that $\mathsf{Open}(c, r) = x, \mathsf{Open}(c, r') = x'$. Hence $A_1(r-r') = 0$ and $A_2(r-r') = x'-x \neq 0 \Rightarrow r \neq r'$. Since $\|r - r'\| \leq 2B_r$, $r - r'$ is a solution to $\mathsf{MSIS}(q, N, d, 2d+1, 2B_r, A_1)$

Next consider the relaxed opening, namely that $\mathcal{A}$ outputs elements c, x, x', r, r', f, f' with valid messages $x \neq x'$ and randomness r, r' such that $\mathsf{Open}(c, r, f) = x, \mathsf{Open}(c, r', f') = x'$. Hence $fc_1 = A_1 r, f'c_1 = A_1 r', fc_2 = A_2 r + fx, f'c_2 = A_2 r + f'x$. By eliminating c_1, c_2, we get $A_1(f'r - fr') = 0, A_2(f'r - fr') = ff'(x' - x)$. Since ff' are invertible $ff'(x' - x) \neq 0$ and thus $f'r - fr' \neq 0$. But f, f' are differences of $\|\cdot\|_1 = 60$ terms and have therefore $\|\cdot\|_2-$norm at most $2\sqrt{60}$. Together with $\|r\|_2, \|r'\|_2 \leq B_r$ we get $\|f'r - fr'\| \leq 4\sqrt{60}B_r$ and therefore $f'r - fr'$ is a solution to $\mathsf{MSIS}(q, N, d, 2d+1, 4\sqrt{60}B_r, A_1)$.

Finally, consider an adversary $\mathcal{A}$ that breaks the hiding property, i.e. can find a message $x \in R_q$ such that he can distinguish a commitment $\mathsf{Com}(x, r)$ (for a random secret $r \in S_{\infty,\beta}^{2d+1}$) from a uniform sample $u \in R_q^{d+1}$. Let $A_i' = (B_i, b_i)$ with b_i the last column of A_i. Let $B' = (B_1^T, B_2^T)^T$ and $A = (B', B)$ for a suitable matrix B. Then B is invertible, since it is of the form $\begin{pmatrix} b_1 & I_d \\ 1 & 0 \end{pmatrix}$. Now B' is sampled uniformly and hence $B^{-1}B$ has uniform entries. Similarly $B^{-1}(u - (0, x)^T)$ is uniform, but $B^{-1}\mathsf{Com}(0, r) = (B^{-1}B', \mathsf{id}_{d+1})r = B^{-1}B'r_1 + r_2$ is a vector of M-LWE samples with secret $r_1 \in R_q^d$ and the error terms are the entries of $r_2 \in R_q^{d+1}$. Thus, if $\mathcal{A}$ can distinguish u from $\mathsf{Com}(x, r)$, it can distinguish $u - (0, x)^T$ from $\mathsf{Com}(x, r)$ and therefore solve $\mathsf{MLWE}(q, N, d, S_{\infty,\beta})$ in the average case, i.e., if the secret is random.

Definition 1 (M-SIS). $\mathsf{MSIS}(q, N, d, 2d+1, 2B_r, A_1)$ *is the problem to find a non-trivial* $s \in S_{2,2B_r}^{2d+1}$ *with* $A_1 s = 0$.

Definition 2 (M-LWE). $\mathsf{MLWE}(q, N, d, S_{\infty,\beta})$ *is the problem to distinguish samples* $(a, \langle a, s\rangle + e)$ *with* $a \leftarrow R_q^d$ *and* $e \leftarrow S_{\infty,\beta}$ *from uniform samples from* $R_q^d \times R_q^d$ *(for a random secret* $s \leftarrow S_{\infty,\beta}^d$*).*

Tallier ZKPs. We want to briefly recall why and how talliers use ZKPs in [32]. As in the Kryvos framework outlined in Section 2, in [32] each tallier T_k receives from each voter V^i a share of the ballot $\boldsymbol{v}_k^i$, a randomness r_k^i, and a commitment $\mathsf{Com}(\boldsymbol{v}_k^i, r_k^i)$. Each T_k then locally aggregates her shares and the corresponding random coins in parallel. Finally, all talliers jointly add their internal sums and publish the resulting $\mathbb{T}$ and the sum $r = \sum_{k=1}^{n_t}\sum_{i=1}^{n_v} r_k^i$. In contrast to classical commitment schemes, like the Pedersen commitments used in Kryvos, in BDLOP the validity of a commitment depends on the size of the randomness. In particular, the sum of valid commitments might no longer contain a sufficiently small r to provide binding, and the election result cannot be verified. To prevent the randomness from growing too big, the talliers in [32] regularly reduce it. Namely, they aggregate a certain number (e.g. 30) in [32], of shares $\boldsymbol{v}_l' = \sum_{i \in B_l} \boldsymbol{v}_k^i$ (called a bucket B_l of shares), such that the sum $r_l' = \sum_{i \in B_l} r_k^i$ for $B_l = \{30l + 1, \ldots, 30(l+1)\}, 0 \leq l < \lceil n_v/30 \rceil$, of the corresponding random coins is still sufficiently small to provide binding of the corresponding aggregated commitment $\mathsf{Com}(\boldsymbol{v}_l', r_l')$. Since the talliers know their used shares (and their sum), they can produce a new commitment with fresh small randomness $\tilde{r}_l \in S_{\infty,\beta}^{2d+1}$. They can publish this new commitment $\mathsf{Com}(\boldsymbol{v}', \tilde{r}_l)$ together with a ZKP showing that the sum of the corresponding original commitments on the bulletin board and the new commitment contain the same value, namely that $\mathsf{Com}(\boldsymbol{v}_l', r_l') - \mathsf{Com}(\boldsymbol{v}_l', \tilde{r}_l) = \mathsf{Com}(0, r_l' - \tilde{r}_l)$ is a valid commitment to 0. Additionally, they prove that $\mathsf{Com}(\boldsymbol{v}', \tilde{r}_l)$ is valid with small randomness $\tilde{r}_l$. Depending on the overall number of voters, the talliers have to repeat the process several times, e.g., to combine different $\boldsymbol{v}_l'$ into larger sums of shares.

For this specific setup, there exist efficient ZKPs, e.g. in [77] or [78], that, for a given commitment c with randomness $\leq B_r$, prove knowledge of a random coin $\hat{r} \leq B_r'$ such that $2c = A\hat{r}$ (where B_r' depends on

B_r and N). If B'_r can be chosen small enough, Theorem 2 with $\overline{\mathcal{C}}$ replaced by $\{2\}$, guarantees $\hat{r} = r'_l - \tilde{r}_l$. Finally, the efficiency of these talliers' ZKPs can be strongly improved by amortizing over many proofs as in Damgård et al. [79] or del Pino and Lyubashevsky [78].

B Implemented Choice Spaces and Result Functions

For PQKryvos, we use the same choice spaces and result functions as the original Kryvos authors. For completeness, we provide an overview here. In the following, we denote by n_{cand} the number of candidates in an election.

B.1 Choice Spaces and Result Functions with $n_{\mathsf{choices}} = n_{\mathsf{option}}$

We first describe the choice spaces and result functions for the settings where each option corresponds to a unique candidate. That is, in PQKryvos, these choicespace require commitments to n_{option} values. In all these methods, the i-th entry of a ballot corresponds to the number of votes/points the i-th candidate received. Hence, the aggregated tally $\mathbb{T}$ contains a list of the overall points/votes that each candidate received.

Choice Spaces Single-Vote. In a single-vote election, a voter can give only one vote for their preferred candidate. The corresponding choice space is defined as

$$\mathsf{Ch}_{\mathsf{single}} := \Big\{(v_1, \ldots, v_{n_{\mathsf{cand}}}) \Big| v_i \in \{0,1\}, \sum_{i=1}^{n_{\mathsf{cand}}} v_i \in \{0,1\}\Big\}.$$

Multi-Vote. Multi-vote is a generalization of single-vote where a voter can distribute up to n_{votes} votes among n_{cand} candidates, with a maximum of t votes assigned to any candidate. It is captured by the following choice space:

$$\mathsf{Ch}_{\mathsf{multi}}(n_{\mathsf{votes}}, t) := \big\{(v_1, \ldots, v_{n_{\mathsf{cand}}}) \mid \forall i : v_i \in \{0,1,\ldots,t\} \wedge 0 \leq \sum_{i=1}^{n_{\mathsf{cand}}} v_i \leq n_{\mathsf{votes}}\big\}.$$

Borda Count.

As [16], we consider the classical Borda method as well as a variant that the authors of [16] called Borda Tournament Style.

In both settings, voters rank the candidates according to their preference and, based on this ranking, points are assigned to each candidate. Variants of the classical Borda method are used, e.g., for parliamentary elections in Nauru [80] and the Eurovision Song Contest (ESC).

The authors of [16] define choicespaces for these variants by defining a fixed point list $\mathcal{P}$ that contains n_{points} many distinct positive numbers. Then, each assignable rank corresponds to a unique number of points defined via a list $\mathcal{P} \in \mathbb{N}^{n_{\mathsf{points}}}$ with $n_{\mathsf{points}} \leq n_{\mathsf{cand}}$, where in the standard case of a full ranking, $n_{\mathsf{points}} = n_{\mathsf{cand}}$ is the number of candidates and $\mathcal{P}[1]$ is the number of points assigned to the candidate that is ranked first and so on. One can consider variants, where multiple candidates may be assigned the same rank, i.e., the same number of points. In that case, if i candidates are assigned the same rank r, the next lower candidate must start at rank $r + i$.

$$\mathsf{Ch}_{\mathsf{Borda}}(\mathcal{P}) = \Big\{(x_1,\dots,x_{n_{\mathsf{cand}}}) \;\Big|\; \forall i : x_i \in \mathcal{P} \wedge \\ \forall r \in [1, n_{\mathsf{points}}], \forall i \in [2, \sigma(r)] : \\ \nexists j \in [1, n_{\mathsf{cand}}] : x_j = \mathcal{P}[r+i-1] \\ \wedge (r + \sigma(r) \leq n_{\mathsf{points}} \\ \Rightarrow \exists j \in [1, n_{\mathsf{cand}}] : x_j = \mathcal{P}[r+\sigma(r)])\Big\},$$

where $\sigma(r) \in [1, n_{\mathsf{cand}}]$ denotes the number of occurrences of $\mathcal{P}[r]$ in $(x_1, \dots, x_{n_{\mathsf{cand}}})$.

The Borda Tournament Style variant is as follows. Based on the ranking of the candidates by a voter, each candidate receives two points per candidate that is ranked lower, and one point for each other candidate with the same rank. This is captured by the following choice space:

$$\mathsf{Ch}_{\mathsf{BordaTournamentStyle}} = \Big\{(x_1,\dots,x_{n_{\mathsf{cand}}}) \;\Big|\; \forall i : \\ x_i = 2 \cdot |\{j \in [1, n_{\mathsf{cand}}] : x_j < x_i\}| \\ + |\{j \in [1, n_{\mathsf{cand}}] \setminus \{i\} : x_j = x_i|\Big\}.$$

Result Functions Since in the above-described election methods the aggregated tally $\mathbb{T}$ contains a list of the overall votes/points that each candidate received, one can define several methods of computing an election result based on $\mathbb{T} = (k_1, \dots k_{n_{\mathsf{cand}}})$.

Most/Least Votes. In these result functions, all candidates that received the highest respectively lowest amount of votes/points are returned. That is, we set $f_{\mathsf{res}}^{\mathsf{MostVotes}}(\mathbb{T}) = \{i \mid k_i = \max(\mathbb{T})\}$ respectively $f_{\mathsf{res}}^{\mathsf{LeastVotes}}(\mathbb{T}) = \{i \mid k_i = \min(\mathbb{T})\}$.

Threshold Set. In this result function, all candidates who received at least a certain threshold t of votes/points are returned. That is, we set $f_{\mathsf{res}}^{\mathsf{ThresholdSet,t}}(\mathbb{T}) = \{i \mid k_i \geq t\}$.

Best n. This result function reveals the n candidates who received the highest votes/points (without ordering). If multiple candidates are tied for the lowest value belonging to the set of n highest values, all of those candidates are revealed. That is, we set $f_{\mathsf{res}}^{\mathsf{BestN},n}(\mathbb{T}) = \{i \mid \#\{j \mid k_j > k_i\} < n\}$.

B.2 Further Election Methods

The remaining considered election methods do not fit the general description above, and each specifies an individual ballot format/choice space and a corresponding result function. We, therefore, sketch them individually on a high level but refer to [16] for a more algorithmic detailed description.

Condorcet with Smith Set. Condorcet Methods aim to elect a Condorcet winner, i.e., a candidate that beats every other candidate in pairwise comparison. However, such a winner may not always exist. A common relaxation is the Smith set, the smallest set of candidates, in which each member defeats all candidates outside the set. If a Condorcet winner exists, the Smith set consists solely of that candidate.

For encoding Condorcet ballots in an aggregatable format that allows Smith set computation, each ballot is represented as a binary, transitive matrix $A \in \{0,1\}^{n_{\mathsf{cand}} \times n_{\mathsf{cand}}}$, encoding pairwise preferences between candidates. In turn, in PQKryvos, a Condorcet ballot/tally requires a commitment to n_{option}^2 values. The corresponding choice space is

$$\mathsf{Ch}_{\mathsf{Condorcet}} = \Big\{A \in \{0,1\}^{n_{\mathsf{cand}} \times n_{\mathsf{cand}}} \;\Big|\; \forall i \neq j : A_{ij} + A_{ji} = 1, \\ A_{ij} = A_{jk} = 1 \Rightarrow A_{ik} = 1\Big\}.$$

IRV. IRV is a complex voting method, e.g., used in Australia [5] or the US [6]. Each voter submits a (potentially) partial ranking of the n_{cand} candidates. A result is derived from these rankings over multiple rounds, where, in each round, if a candidate is ranked first by an absolute majority of voters, this candidate immediately wins the election. Otherwise, the candidate ranked first by the least number of voters is eliminated. As a result of this elimination, all ballots are edited by removing the eliminated candidate, allowing all of the following candidates to move up in the ranking. This process is then iterated. To handle possible ties, we consider the tie-breaking algorithm used in New South Wales [5], where if ties occur, they are broken by eliminating the candidate who had received the fewest votes in a previous round. If this cannot break a tie, it is broken by lot instead.

Since Kryvos and PQKryvos work on an aggregated tally that still contains information on all submitted rankings, we use the choice space $\mathsf{Ch}_{\mathsf{single}}$ from above, but interpret each choice as a (full or partial) ranking of candidates, yielding a large $n_{\mathsf{choices}} = \sum_{k=1}^{n_{\mathsf{cand}}} \binom{n_{\mathsf{cand}}}{k} \cdot k!$. Hence, for IRV elections, PQKryvos requires commitments to that large number of values.

Majority Judgment. In a Majority Judgment election, a voter can independently assign one of n_{Grades} grades from a pre-defined list to each candidate. A winner is determined based on the best median grade that she received from the total of all voters. The result computation iterates over multiple rounds if two candidates are tied for the best median. In each round, the best obtained median grade is computed. All candidates who currently have a worse median grade are eliminated. Then, one vote for this median grade is removed for every remaining candidate. For the next round, the median grades are updated accordingly to the removal of the single vote. This procedure is repeated until only one candidate, the election winner, is left. We call this a "full" evaluation of a Majority Judgment election (as opposed to a variant where all tied candidates would be revealed)

In order to make Majority Judgment ballots aggregatable and to use an efficient tallying algorithm [81], the authors of Kryvos encode ballots via the following choice space

$$\begin{aligned}\mathsf{Ch}_{\mathsf{MajorityJudgment}} = \Big\{ & A \in \{0,1\}^{n_{\mathsf{cand}} \times n_{\mathsf{Grades}}} \Big| \\ & \forall i \in [1, n_{\mathsf{cand}}], j \in [1, n_{\mathsf{Grades}}] : \\ & A_{ij} \in \{0,1\} = 1 \\ & \wedge \forall i \in [1, n_{\mathsf{cand}}] : \sum_{j \in [1, n_{\mathsf{Grades}}]} A_{ij} = 1 \Big\},\end{aligned}$$

requiring commitments to $n_{\mathsf{cand}} \cdot n_{\mathsf{Grades}}$ values.

C Security Proof for PQKryvos

In this section, we formally prove that PQKryvos is a verifiable e-voting protocol that achieves vote privacy, and even public tally-hiding. For this purpose, we first formally model PQKryvos within the established KTV framework [12, 14, 75], which we adapt to also consider quantum adversaries. We then analyze the aforementioned security properties within that framework. For public tally-hiding, we recall the definition from [16] in Appendix C.3, which is also formalized within the KTV framework.

C.1 Computational Model of PQKryvos

Computational Framework. To formally model a *protocol*, we follow Kuesters et al. [12, 14] and start with the notion of a *process* $\pi = p_1 \| \cdots \| p_l$, which is a set of probabilistic polynomial-time interactive Turing machines (*programs*) $p_1, \ldots, p_l$. The programs p_i are connected via named directional tapes (*channels*); depending on the direction, we call them input resp. output channel of the program. A process may have internal channels (input/output channels between its programs) and external input or output channels (that are not connected internally). Two programs are connected if they share an input and output channel of

the same name. If all programs within a process π share the same security parameter 1^κ, we also denote π as $\pi^{(\kappa)}$. We further assume that a process defines a set of runs and a probability distribution among them (recall that programs are probabilistic). Each run contains a description of the corresponding process, the security parameter, and all used random coins.

During a process run, at any given time, only one program is considered active, i.e., it may execute some computation and send a message (e.g., to another program) via one of its output channels. If a message is sent to another program this way, then that program becomes the active program. A process contains a *master program*, which is the first program to be activated and which is activated if the active program did not produce output (and hence, did not activate another program). If the master program is active but does not produce output, a run stops.

If two processes π_1,π_2 share connected external input and output channels, we write $\pi_1 \| \pi_2$ for the connected process (renaming internal channels if necessary).

A *protocol* P is defined by a set Σ of *protocol participants*. For each $a \in \Sigma$, a program $\hat{p}_a$ (a's *honest program*) is defined, which is the program that a is supposed to run if it behaves honestly. The honest programs $\{\hat{p}_a \mid a \in \Sigma\}$ are mutually connected by input and output channel and, for $\Sigma = \{a_1, \ldots, a_l\}$ we denote the process $\hat{p}_{a_1} \| \ldots \| \hat{p}_{a_l}$ by $\hat{\pi}_\mathsf{P}$.

The process $\hat{\pi}$ is assumed to run with a participant $\mathcal{A}$, the *adversary*. $\mathcal{A}$ can run an aribtrary probabilistic polynomial-time quantum program $p_\mathcal{A}$, which is connected to each $\hat{p}_a$ with $a \in \Sigma$. For any program $p_\mathcal{A}$ of $\mathcal{A}$, we call $\hat{\pi} \| p_\mathcal{A}$ an *instance* of P. A run r of the protocol P with adversary program $p_\mathcal{A}$ is a run of the process $\hat{\pi}_\mathsf{P} \| p_\mathcal{A}$. A *property* γ of protocol P is a subset of the set of all runs of P. By $\neg\gamma$, we denote the complement of γ.

We specify the honest programs $\hat{p}_a$ for $a \in \Sigma$ in a way that allows $\mathcal{A}$ to send a special message `corrupt` to a, which - if a accepts `corrupt` - leads to a revealing all or some of its internal state to $\mathcal{A}$. From then on, a is controlled by $\mathcal{A}$. We say that participant a is *honest in protocol run* r if it does not accept `corrupt` in this run r. If a protocol participant $a \in \Sigma$ never accepts `corrupt` (in any protocol run), we say that a is *honest*. If all process in P (including $p_\mathcal{A}$) share the same security parameter 1^κ, we denote P as $\mathsf{P}^{(\kappa)}$.

Modeling of PQKryvos Within the computational framework described above, we model PQKryvos as a protocol $\mathsf{P}_\mathsf{PQKryvos} := \mathsf{P}_\mathsf{PQKryvos}(n_v, n_t, \mathsf{Ch}, f_\mathsf{res}, \mu)^{(\kappa)}$, where n_v denotes the number of voters, n_t denotes the number of talliers, Ch denotes the used choice space, f_res denotes the used result function, and μ denotes a probability distribution on Ch according to which honest voters vote (modelling that μ is known to $\mathcal{A}$). Besides the n_v voters and the n_t talliers (with honest programs derived in the obvious way from the protocol description in Section 2), the set of participants of $\mathsf{P}_\mathsf{PQKryvos}$ contains a bulletin board BB and a scheduler S, the latter of which is a virtual entity that subsumes the voting authority Auth and whose honest program $\hat{p}_\mathsf{S}$ acts as the master program of the process $\hat{\pi}_\mathsf{P} \| p_\mathcal{A}$ in every instance of $\mathsf{P}_\mathsf{PQKryvos}$. In particular, S schedules all other agents in a run according to the protocol phases. For the security analysis of PQKryvos, we assume that both BB and S are honest. We note that, in order to mitigate trust assumptions for BB, one could instantiate BB with an accountable bulletin board and/or in a distributed fashion [76, 82]. Moreover, we assume *static corruption*. That is, we assume that in any run of $\mathsf{P}_\mathsf{PQKryvos}$, protocol participants only accept `corrupt` upon initialization.

C.2 Verifiability

We analyze the verifiability level of PQKryvos within the established KTV framework. First, we recall a slightly simplified version of the verifiability definition proposed in [75]. We note that, in the KTV definition, verifiability is defined in both a symbolic and a computational fashion. In the following, we only consider the computational definition and, hence, consider protocols that depend on a security parameter 1^κ. On a high level, the definition captures that there is only a small probability of an incorrect result being published without any observed misbehavior observed.

Verifiability Framework. Let $\mathsf{P} = \mathsf{P}^{(\kappa)}$ be a protocol in the sense of Appendix C.1. In order to define verifiability, the KTV framework assumes an honest protocol participant J called the *judge*. J can accept

or reject a protocol run by writing `accept` or `reject` on a dedicated channel. The honest program $\hat{p}_{\mathsf{J}}$ of J consists in performing certain checks specific to the considered protocol.

In the KTV framework, verifiability is formalized via a property γ of P, which is called a *goal*. That is, γ is a subset of the set of all runs of P. We denote by $\Pr[(\hat{\pi}_{\mathsf{P}} \| p_{\mathcal{A}})^{(\kappa)} \mapsto \neg\gamma, (\mathsf{J}\colon \texttt{accept})]$ the probability that a run of the protocol P with adversary $p_{\mathcal{A}}$ does not belong to γ but J nevertheless writes `accept` on the dedicated channel.

Definition 3. *The goal γ is verifiable by the judge J in the protocol $\mathsf{P} = \mathsf{P}^{(\kappa)}$ if for any adversary program $p_{\mathcal{A}}$, the probability $\Pr[(\hat{\pi}_{\mathsf{P}} \| p_{\mathcal{A}})^{(\kappa)} \mapsto \neg\gamma, (\mathsf{J}\colon \texttt{accept})]$ is negligible as a function in κ.*[15,16]

The goal $\gamma(\varphi)$. Following [12, 16], we define for $\mathsf{PQKryvos}$ the goal $\gamma(\varphi)$, which captures the following requirement: if S, BB, and J are honest (which is expressed through φ), then in every run of $\mathsf{P}_{\mathsf{PQKryvos}}$, the announced election result must be consistent with the actual votes cast by the voters that are honest in this run (together with an arbitrary multiset of valid votes corresponding in number to the dishonest participants).

Formally, the goal $\gamma(\varphi)$ is defined as follows for an arbitrary voting method parametrized by a choice space Ch and a result function f_{res}:

Definition 4. *Let $\mathsf{P} = \mathsf{P}_{\mathsf{PQKryvos}}$. For a protocol participant $a \in \Sigma$, let $\mathsf{hon}(a)$ denote that a is honest (in all runs of P). Let $\varphi = \mathsf{hon}(\mathsf{S}) \wedge \mathsf{hon}(\mathsf{BB}) \wedge \mathsf{hon}(\mathsf{J})$. Let I_h and I_d denote the set of honest and dishonest voters, respectively, in a given run of P (recall that we assume static corruption, i.e., the set of honest/dishonest parties is determined at the beginning of each protocol run). Then, $\gamma(\varphi)$ consists of all those runs of $\mathsf{PQKryvos}$ where either*

- *φ does not hold, or*
- *φ holds and there exist (valid) choices $m_i \in \mathsf{Ch}$ for $i \in I_d$ such that the election result equals $f_{\mathsf{res}}(\sum_{i \in I_h \cup I_d} \boldsymbol{v}_i)$, where the $\boldsymbol{v}_i \in \mathsf{Ch}$ for $i \in I_h$ are the honest voters' choices.*

Remark 3. For brevity this is a simplified version of the goal described in [12]. In particular, this model abstracts from voter-side checks (e.g., verifying that their encrypted ballots appear on BB) and does not distinguish between voters and voting devices. We assume throughout that voters perform all required checks and that ballots are cast as intended. Some standard mechanisms for ensuring the latter, such as Benaloh challenges, are orthogonal to and fully compatible with $\mathsf{PQKryvos}$; incorporating them would result in a fully end-to-end verifiable system. Detailed verifiability analyses that explicitly account for these aspects have been carried out for the Ordinos [17] and the Helios protocol [83] and the corresponding results apply analogously in our setting.

Verifiabiltiy Proof Here, we give a full proof of Theorem 5 $\mathsf{PQKryvos}$ from Section 6:

Proof. Assume that φ holds, i.e., S, BB, and J are honest (otherwise there is nothing to prove). We need to show that, if the final result res does not equal $f_{\mathsf{res}}(\sum_{i \in I} \boldsymbol{v}_i)$, where $(\boldsymbol{v}_i)_{i \in I}$ consists of all honest voters' choices and at most one (valid) choice per dishonest voter, then the judge J writes `reject` except with negligible probability.

First note that (up to a negligible error probability) J sends `reject` if an honest voter's vote is discarded before the tallying phase. Indeed, since honest voters create valid ballots, by completeness of Π, their

[15] The original definition from [75] also requires that J accepts, at least, if an adversary does not corrupt any party. This can be seen as a sanity check of the overall protocol and is usually easy to check. For brevity, we omit this condition here.

[16] The original definition also introduces a tolerance term $\delta \in [0, 1]$ and only requires this probability to be asymptotically bounded by $\delta + \kappa^{-c}$ for every $c > 0$. This captures that, e.g., the required checks by J might not always be performed. For instance, in e-voting protocols, these checks might include voters checking that their ballot appears on the bulletin board, which typically not all voters do. For the security analysis of $\mathsf{PQKryvos}$ we abstract away from this by setting $\delta = 0$ and refer to, e.g., [17] for a more detailed analysis.

NIZKPoK π_{ballot} is valid. Moreover, their ballots are well-formed and their identity i belongs to **id**. Finally, by the IND-CCA2 security of E, the probability that an honest voter's ballot contains an identitcal ciphertext to a previously submitted ballot is negligible (assuming that the number of eligible voters is polynomially bounded). Hence, honest voters' votes are not discarded by the honest BB but appended to the public list of ballots on BB up to a negligible error probability. That means that the only way of dicarding an honest voter's vote before the tallying phase would be that a corrupted tallier publishes a complaint on BB consisting of a wrong decryption result as well as a claimed zero-knowledge proof π_{dec} of correct decryption. However, by soundness of π_{dec}, this will be detected by J (up to the negligible soundness error of π_{dec}) who, in turn, writes `reject`.

Moreover, at the beginning of the tallying phase, the eligibility and re-voting checks of the honest BB ensure that no more than one choice per dishonest voter is considered. Furthermore, up to a negligible error probability, also these ballots must contain commitments to valid votes, since otherwise, by the soundness of Π, the associated ballot validity proof π_{ballot} would not be valid and, hence, the ballots would be discarded by the honest BB.

Now, let I_{valid} denote the set of all these valid ballots that are taken into account for the tallying phase. As argued above, I_{valid} contains all honest voters' ballots and at most one (valid) vote per dishonest voter. Since the commitment scheme C is additively homomorphic and computationally binding, the aggregation c of all commitments from ballots in I_{valid} is a valid commitment to $\sum_{i\in I_{\mathsf{valid}}} \boldsymbol{v}_i$ (up to a negligible error probability). Finally, if the final result res does not equal $f_{\mathsf{res}}(\sum_{i\in I_{\mathsf{valid}}} \boldsymbol{v}_i)$, then the soundness of Π implies that, up to a negligible error, the NIZKPoK π_{result} is not valid and, hence, J sends `reject`.

Remark 4. While the proof of Theorem 5 ist mostly the same as the verifiability proof in [16], we point out two minor differences. First, our analysis also accounts for the (negligible) decryption error that many pq-secure encryption schemes introduce. Moreover, the judge J in our theorem does not need to perform eligibility checks or check validity of π_{ballot}, since BB, which is assumed to be honest, already performs these checks.

C.3 Privacy and Public Tally-Hiding

We analyze the privacy of $\mathsf{PQKryvos}$ according to the definition from the KTV framework [14], briefly recalled here. This definition allows the exact computation of an e-voting protocol's privacy level based on the choice space Ch, the number $n^{\mathsf{hon}}_{\mathsf{voters}}$ of non-abstaining honest voters, and the distribution μ of honest votes. It can also be expressed as a more intuitive indistinguishability game, capturing an adversary's inability to determine whether a given honest voter $\mathsf{V}_{\mathsf{obs}}$ (the *voter under observation*) voted for $\boldsymbol{v}_0 \in \mathsf{Ch}$ or $\boldsymbol{v}_1 \in \mathsf{Ch}$. This game-based formulation has been used in prior work, e.g., [16, 17, 76], and is the approach we adopt here. Additionally, we evaluate the public tally-hiding of $\mathsf{PQKryvos}$ following the definition of [16], which extends the original KTV privacy notion.

Privacy Framework. Let $\mathsf{P} = \mathsf{P}(n_v, \mathsf{Ch}, f_{\mathsf{res}}, \mu)^{(\kappa)}$, be an arbitrary voting protocol (e.g., $\mathsf{P}_{\mathsf{PQKryvos}}$), where n_v denotes the number of voters, Ch denotes the used choice space, f_{res} denotes the used result function, and μ denotes a probability distribution on Ch according to which honest voters vote.

For a voter under observation $\mathsf{V}_{\mathsf{obs}}$ and a choice $\boldsymbol{v} \in \mathsf{Ch}$ we denote by $(\hat{p}_{\mathsf{V}_{\mathsf{obs}}}(\boldsymbol{v})\|\pi^*\|p_{\mathcal{A}})^{(\kappa)}$ the instance of P where $\mathsf{V}_{\mathsf{obs}}$ runs its honest program voting for $\boldsymbol{v}$, π^* is the composition of programs of all other participants in P, and $p_{\mathcal{A}}$ is the adversary's program.

Let $\Pr[(\hat{p}_{\mathsf{V}_{\mathsf{obs}}}(\boldsymbol{v})\|\pi^*\|p_{\mathcal{A}})^{(\kappa)} \mapsto 1]$ denote the probability that the adversary writes the output 1 on some dedicated channel in a run of $(\hat{p}_{\mathsf{V}_{\mathsf{obs}}}(\boldsymbol{v})\|\pi^*\|p_{\mathcal{A}})^{(\kappa)}$, where the probability is taken over the random coins used by the parties in $(\hat{p}_{\mathsf{V}_{\mathsf{obs}}}(\boldsymbol{v})\|\pi^*\|p_{\mathcal{A}})^{(\kappa)}$.

Privacy is then defined via the security game $\mathbb{E}(\mathcal{A}, \mathsf{P}, \kappa, \mathsf{V}_{\mathsf{obs}})$ (cf. Figure 3), where we often write $\mathbb{E}(\mathcal{A})$ or $\mathbb{E}$ when all other parameters are clear from the context.

Note that we cannot expect the winning probability $\Pr[\mathbb{E}(\mathcal{A}, \mathsf{P}, \kappa, \mathsf{V}_{\mathsf{obs}}) \mapsto 1]$ (taken over the random coins used in $(\hat{p}_{\mathsf{V}_{\mathsf{obs}}}(\boldsymbol{v})\|\pi^*\|p_{\mathcal{A}})$ and the random choice of b in step 2) to be negligible - e.g., an adversary that corrupts all but one voter $\mathsf{V}_{\mathsf{obs}}$ can trivially learn b from the election result. Hence, privacy of P is formalized via a comparison of this winning probability with

$\mathbb{E}(\mathcal{A}, \mathsf{P} = \mathsf{P}(n_v, \mathsf{Ch}, f_{\mathsf{res}}, \mu)^{(\kappa)}, \kappa, \mathsf{V}_{\mathsf{obs}})$**:**

1. $\mathcal{A}$ chooses $\boldsymbol{v}_0, \boldsymbol{v}_1 \in \mathsf{Ch}$ and sends them to the challenger.
2. Choose a bit $b \leftarrow \{0,1\}$ uniformly at random.
3. Execute the instance $(\hat{p}_{\mathsf{V}_{\mathsf{obs}}}(\boldsymbol{v}_b) \| \pi^*_{\mathsf{P}} \| p_{\mathcal{A}})$ of P.
4. $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$.
5. Output 1 if $b' = b$, and 0 otherwise.

Fig. 3: Security game defining privacy.

the winning probability in a security game with an *ideal* protocol $\mathsf{P}^{\mathsf{ideal}}(n_v, n^{\mathsf{hon}}_{\mathsf{voters}}, \mathsf{Ch}, f_{\mathsf{res}}, \mu)$ that only reveals the election result but nothing else (cf. Figure 4). Privacy is then defined w.r.t a specific set of adversaries that we call the *admissible adversaries* (e.g., for PQKryvos, these are adversaries that do not corrupt BB or S, corrupt at most all but one tallier, and at most $n^{\mathsf{hon}}_{\mathsf{voters}}$ voters):

Definition 5. *A voting protocol* $\mathsf{P} = \mathsf{P}(n_v, \mathsf{Ch}, f_{\mathsf{res}}, \mu)^{(\kappa)}$ *achieves* (ideal) priacy *if for all admissible adversaries* $\mathcal{A}$ *on* P *there is an adversary* $\mathcal{A}_{\mathsf{ideal}}$ *on* $\mathsf{P}^{\mathsf{ideal}} = \mathsf{P}^{\mathsf{ideal}}(n_v, n^{\mathsf{hon}}_{\mathsf{voters}}, \mathsf{Ch}, f_{\mathsf{res}}, \mu)$ *such that*

$$\left|\mathsf{Pr}[\mathbb{E}((\mathcal{A}, \mathsf{P}, \kappa, \mathsf{V}_{\mathsf{obs}})) \mapsto 1] - \mathsf{Pr}[\mathbb{E}((\mathcal{A}_{\mathsf{ideal}}, \mathsf{P}^{\mathsf{ideal}}, \kappa, \mathsf{V}_{\mathsf{obs}})) \mapsto 1]\right|$$

is negligible as a function in κ*.*

Public Tally-Hiding. Public tally-hiding [16] is defined based on Definition 5 by specifying two separate privacy notions (*public* and *internal* privacy), each with its own set of admissible adversaries, A_{pub} and A_{int}, and related ideal voting protocol. In order to be applicable, we assume the considered voting protocol to n_t talliers among its participants. Then, public privacy captures that if all talliers are honest, then nothing beyond the election result is revealed. Internal privacy captures that if at least one tallier is honest, then nothing beyond the aggregated tally is revealed. Formally:

Definition 6. *A voting protocol* $\mathsf{P} = \mathsf{P}(n_v, n_t\mathsf{Ch}, f_{\mathsf{res}}, \mu)^{(\kappa)}$ *achieves* public tally-hiding *if the following two conditions hold true:*

1. *For all admissible adversaries* $\mathcal{A} \in A_{\mathsf{pub}}$ *on* P *there is an adversary* $\mathcal{A}_{\mathsf{ideal}}$ *on* $\mathsf{P}^{\mathsf{ideal}} = \mathsf{P}^{\mathsf{ideal}}(n_v, n^{\mathsf{hon}}_{\mathsf{voters}}, \mathsf{Ch}, f_{\mathsf{res}}, \mu)$ *such that*

$$\left|\mathsf{Pr}[\mathbb{E}((\mathcal{A}, \mathsf{P}, \kappa, \mathsf{V}_{\mathsf{obs}})) \mapsto 1] - \mathsf{Pr}[\mathbb{E}((\mathcal{A}_{\mathsf{ideal}}, \mathsf{P}^{\mathsf{ideal}}, \kappa, \mathsf{V}_{\mathsf{obs}})) \mapsto 1]\right|$$

   is negligible as a function in κ *(*public privacy*).*
2. *For all admissible adversaries* $\mathcal{A} \in A_{\mathsf{int}}$ *on* P *there is an adversary* $\mathcal{A}_{\mathsf{ideal}}$ *on* $\mathsf{P}^{\mathsf{ideal}} = \mathsf{P}^{\mathsf{ideal}}(n_v, n^{\mathsf{hon}}_{\mathsf{voters}}, \mathsf{Ch}, \mathsf{id}, \mu)$ *such that*

$$\left|\mathsf{Pr}[\mathbb{E}((\mathcal{A}, \mathsf{P}, \kappa, \mathsf{V}_{\mathsf{obs}})) \mapsto 1] - \mathsf{Pr}[\mathbb{E}((\mathcal{A}_{\mathsf{ideal}}, \mathsf{P}^{\mathsf{ideal}}, \kappa, \mathsf{V}_{\mathsf{obs}})) \mapsto 1]\right|$$

   is negligible as a function in κ *(*internal privacy*).*

Remark 5. If $f_{\mathsf{res}} = \mathsf{id}$, i.e., in the non-tally-hiding case, public and internal privacy are the same apart from the considered admissible adversaries. In the usual case where $A_{\mathsf{pub}} \subseteq A_{\mathsf{int}}$, this notion is the same as ideal privacy w.r.t. adversaries from A_{int}.

Privacy Proof. Here, we give a full proof of Theorem 6 from Section 6:

Proof. First, we fix some notation. Recall that for $\mathbb{E}(\mathcal{A}, \mathsf{P}, \kappa, \mathsf{V}_{\mathsf{obs}})$ we always consider runs of the form $(\hat{p}_{\mathsf{V}_{\mathsf{obs}}}(\boldsymbol{v}_b)\|\pi^*_{\mathsf{P}}\|p_{\mathcal{A}})$. In the following, we denote by $\hat{p}_{\mathsf{H}}(\boldsymbol{v})$ the composition of all honest programs including the program of the voter under observation $\mathsf{V}_{\mathsf{obs}}$ when voting for $\boldsymbol{v}$. All remaining programs are subsumed by the adversarial process $\pi_{\mathcal{A}}$, so we write $\hat{p}_{\mathsf{V}_{\mathsf{obs}}}(\boldsymbol{v})\|\pi^*\|p_{\mathcal{A}}$ as $\hat{p}_{\mathsf{H}}(\boldsymbol{v})\|\pi_{\mathcal{A}}$.

Now, we prove Theorem 6 via two separate sequences of games, one for internal privacy, and one for public privacy. We start with *internal privacy*:

Let $\mathcal{A} \in A_{\mathsf{int}}$, i.e., $\mathcal{A}$ does not corrupt BB or S, corrupts at most $n^{\mathsf{hon}}_{\mathsf{voters}}$ voters, and corrupts at most all but one tallier. W.l.o.g. we assume that T_1 is not corrupted by $\mathcal{A}$. That is, $\hat{p}_{\mathsf{H}}(\boldsymbol{v})$ consists of $\hat{p}_{\mathsf{BB}}, \hat{p}_{\mathsf{S}}, \hat{p}_{\mathsf{T}_1}$, and $\hat{\pi}_{\mathsf{HV}}(\boldsymbol{v})$, where the latter is the composition of all $n^{\mathsf{hon}}_{\mathsf{voters}}$ voters' programs (including $\mathsf{V}_{\mathsf{obs}}$ voting for $\boldsymbol{v}$). We start our sequence of games with $\mathbb{E}_0$ in which $\hat{p}_{\mathsf{H}}$ is as specified by $\mathsf{PQKryvos}$. The game hopping steps then work as follows:

1. $\mathbb{E}_0 \to \mathbb{E}_1$: We define $\mathbb{E}_1$ to be the same as $\mathbb{E}_0$ except for the following change to $\hat{p}_{\mathsf{T}_1}$:
   If T_1 finds that an honest voter's ciphertext ct_1 (encrypted with T_1's public key) decrypts to an invalid opening for the corresponding commitment, then T_1 aborts.
   Since an honest voter always encrypts a valid opening, the difference in any admissible adversary $\mathcal{A}$'s winning probability for the two games $\mathbb{E}_0(\mathcal{A})$ and $\mathbb{E}_1(\mathcal{A})$ is bounded by the negligible decryption error probability of E. That is,
   $$|\mathsf{Pr}[\mathbb{E}_0(\mathcal{A}) \mapsto 1] - \mathsf{Pr}[\mathbb{E}_1(\mathcal{A}) \mapsto 1]|$$
   is negligible.
2. $\mathbb{E}_1 \to \mathbb{E}_2$: We define $\mathbb{E}_2$ to be the same as $\mathbb{E}_1$ except for two changes:
   First, we change $\hat{\pi}_{\mathsf{HV}}$ (including $\hat{p}_{\mathsf{V}_{\mathsf{obs}}}$) as follows: Instead of encrypting their committed vote shares under T_1's public key, honest voters encrypt dummy values (e.g., 0) and send them to BB. Second, we change $\hat{p}_{\mathsf{T}_1}$ as follows: T_1 internally receives the correct openings for the honest voters' committed vote shares (inside of the simulator which subsumes all honest parties) and ignores the encrypted dummy values.
   Since E is IND-CCA2-secure against ppt quantum adversaries (i.e., $\mathcal{A}$ cannot distinguish between encryptions of vote shares and encryptions of dummy values) and since duplicate ciphertexts are removed by the honest BB, we get that
   $$|\mathsf{Pr}[\mathbb{E}_1(\mathcal{A}) \mapsto 1] - \mathsf{Pr}[\mathbb{E}_2(\mathcal{A}) \mapsto 1]|$$
   is negligible.[17]
3. $\mathbb{E}_2 \to \mathbb{E}_3$: We define $\mathbb{E}_3$ to be the same as $\mathbb{E}_2$ except for the following change to $\hat{\pi}_{\mathsf{HV}}$:
   Instead of using Prove_{Π} to create NIZKPoKs π_{ballot} of ballot validity, honest voters invoke the ZKP simulator for Π to create simulated NIZKPoKs and publish these as part of their ballot.
   Since Π is computationally zero-knowledge against ppt quantum adversaries, i.e., $\mathcal{A}$ cannot distinguish between real and simulated NIZKPoKs, we get that
   $$|\mathsf{Pr}[\mathbb{E}_2(\mathcal{A}) \mapsto 1] - \mathsf{Pr}[\mathbb{E}_3(\mathcal{A}) \mapsto 1]|$$
   is negligible.
4. $\mathbb{E}_3 \to \mathbb{E}_4$: We define $\mathbb{E}_4$ to be the same as $\mathbb{E}_3$ except for the following change to $\hat{p}_{\mathsf{T}_1}$:
   If T_1 receives an invalid opening for one of its ciphertexts, then instead of computing π_{dec}, it uses the ZKP simulator for π_{dec} to create a simulated NIZKP and publishes it alongside the invalid opening.
   Since π_{dec} achieves computational zero-knowledge against ppt quantum adversaries, i.e., $\mathcal{A}$ cannot distinguish between real and simulated NIZKPs, we get that
   $$|\mathsf{Pr}[\mathbb{E}_3(\mathcal{A}) \mapsto 1] - \mathsf{Pr}[\mathbb{E}_4(\mathcal{A}) \mapsto 1]|$$
   is negligible.

[17] Note that the removal of ciphertext duplicates prevents $\mathcal{A}$ from simply copying the honest voters' encryptions of dummy values which would be noticeable as T_1 would either abort or the result would be skewed.

5. $\mathbb{E}_4 \to \mathbb{E}_5$: We define $\mathbb{E}_5$ to be the same as $\mathbb{E}_4$ except for two changes regarding $\hat{\pi}_{\mathsf{HV}}$:
   First, instead of computing commitments to (shares of) their actual votes, the honest voters (except for $\mathsf{V}_{\mathsf{obs}}$) publish commitments to the zero-vector. Second, for all other honest voters, $\mathsf{V}_{\mathsf{obs}}$ samples a vote $\boldsymbol{v} \xleftarrow{\mu} \mathsf{Ch}$. It then computes the aggregated honest voters vote $\boldsymbol{v}'$ as the sum of all these sampled votes and its own vote $\boldsymbol{v}_b$. Then, $\mathsf{V}_{\mathsf{obs}}$ computes and publishes its commitments to the shares of $\boldsymbol{v}'$. Since all votes are homomorphically aggregated in the tallying phase, it does not matter how the honest voters' aggregated choice is split among the homomorphic commitments.
   Moreover, since C is computationally hiding against ppt quantum adversaries, since duplicate ciphertexts are removed by the honest BB, and since no proper subset of the n_t talliers can reconstruct a single voter's vote, this change can only negligibly affect an admissible adversary's winning probability, i.e.,

$$|\mathsf{Pr}[\mathbb{E}_4(\mathcal{A}) \mapsto 1] - \mathsf{Pr}[\mathbb{E}_5(\mathcal{A}) \mapsto 1]|$$

   is negligible.
6. $\mathbb{E}_5 \to \mathbb{E}_6$: We define $\mathbb{E}_6$ to be the same as $\mathbb{E}_5$ except for the following change to $\hat{p}_{\mathsf{V}_{\mathsf{obs}}}$:
   Instead of sampling the other honest voters's votes according to μ (as described for $\mathbb{E}_5$), $\mathsf{V}_{\mathsf{obs}}$ invokes the ideal voting protocol $\mathsf{P}^{\mathsf{ideal}}(n^{\mathsf{hon}}_{\mathsf{voters}}, n^{\mathsf{hon}}_{\mathsf{voters}}, \mathsf{Ch}, \mathsf{id}, \mu)[\boldsymbol{v}_b]$ to obtain the honest voters' aggregated vote $\boldsymbol{v}'$.
   Since the ideal voting protocol $\mathsf{P}^{\mathsf{ideal}}(n^{\mathsf{hon}}_{\mathsf{voters}}, n^{\mathsf{hon}}_{\mathsf{voters}}, \mathsf{Ch}, \mathsf{id}, \mu)[\boldsymbol{v}]$ obtains $\boldsymbol{v}'$ just in the same way as $\mathsf{V}_{\mathsf{obs}}$ previously did, this change does not affect an adversary's winning probability, i.e.,

$$|\mathsf{Pr}[\mathbb{E}_5(\mathcal{A}) \mapsto 1] - \mathsf{Pr}[\mathbb{E}_6(\mathcal{A}) \mapsto 1]| = 0.$$

The previous steps have shown that for any adversary $\mathcal{A} \in A_{\mathsf{int}}$ on $\mathsf{PQKryvos}$, there is an adversary $\mathcal{A}_6$ on $\mathbb{E}_6$ such that the difference

$$|\mathsf{Pr}[\mathbb{E}(\mathcal{A}, \mathsf{P}, \kappa, \mathsf{V}_{\mathsf{obs}}) \mapsto 1] - \mathsf{Pr}[\mathbb{E}_6(\mathcal{A}_6) \mapsto 1]|$$

is negligible. However, note that $\mathcal{A}_6$ does not depend on the honest voters' actual votes except for the aggregated sum of all votes that the talliers internally compute, which is computed in the same way as in $\mathsf{P}^{\mathsf{ideal}}(n_v, n^{\mathsf{hon}}_{\mathsf{voters}}, \mathsf{Ch}, \mathsf{id}, \mu)$. Hence, by setting $\mathcal{A}_{\mathsf{ideal}} = \mathcal{A}_6$, we see that

$$\left|\mathsf{Pr}[\mathbb{E}(\mathcal{A}, \mathsf{P}, \kappa, \mathsf{V}_{\mathsf{obs}}) \mapsto 1] - \mathsf{Pr}[\mathbb{E}((\mathcal{A}_{\mathsf{ideal}}, \mathsf{P}^{\mathsf{ideal}}, \kappa, \mathsf{V}_{\mathsf{obs}})) \mapsto 1]\right|$$

is negligible, i.e., $\mathsf{PQKryvos}$ achieves internal privacy.

The proof for *public privacy* works very similarly. Recall that here, we assume all talliers to be honest. We, hence, simplify the notation by assuming a single tallier T whose honest program $\hat{p}_{\mathsf{T}}$ subsumes the honest programs of all n_t talliers. That is, $\hat{p}_{\mathsf{H}}(\boldsymbol{v})$ consists of $\hat{p}_{\mathsf{BB}}, \hat{p}_{\mathsf{S}}, \hat{p}_{\mathsf{T}}$, and $\hat{\pi}_{\mathsf{HV}}(\boldsymbol{v})$, where the latter is the composition of all $n^{\mathsf{hon}}_{\mathsf{voters}}$ voters' programs (including $\mathsf{V}_{\mathsf{obs}}$ voting for $\boldsymbol{v}$). Again, we start the sequence of games with $\mathbb{E}_0$ in which $\hat{p}_{\mathsf{H}}$ is as specified by $\mathsf{PQKryvos}$. The first game hopping steps up to $\mathbb{E}_4$ are as for internal privacy (replacing T_1 with T where necessary). Recall that

$$|\mathsf{Pr}[\mathbb{E}_0(\mathcal{A}) \mapsto 1] - \mathsf{Pr}[\mathbb{E}_4(\mathcal{A}) \mapsto 1]|$$

is negligible. Now, for public privacy, we proceed as follows:

5. $\mathbb{E}_4 \to \mathbb{E}_5$: We define $\mathbb{E}_5$ to be the same as $\mathbb{E}_4$ except for the following change regarding $\hat{p}_{\mathsf{T}}$:
   Instead of using Prove_{Π} to create NIZKPoKs π_{result} of correct result computation, T invokes the ZKP simulator for Π to create a simulated NIZKPoK and publishes it in the end of the tallying phase.
   Since Π is computationally zero-knowledge against ppt quantum adversaries, i.e., $\mathcal{A}$ cannot distinguish between real and simulated NIZKPoKs, we get that

$$|\mathsf{Pr}[\mathbb{E}_4(\mathcal{A}) \mapsto 1] - \mathsf{Pr}[\mathbb{E}_5(\mathcal{A}) \mapsto 1]|$$

   is negligible.

6. $\mathbb{E}_5 \to \mathbb{E}_6$: We define $\mathbb{E}_6$ to be the same as $\mathbb{E}_5$ except for the following change regarding $\hat{p}_{\mathsf{T}}$:
   T starts by computing $\boldsymbol{v}_d$ as the aggregated sum of all votes cast by dishonest voters (since T subsumes all n_t talliers, it can obtain the individual votes of dishonest voters by combining their ciphertexts). Then, for all honest voters (except for $\mathsf{V}_{\mathsf{obs}}$), T samples a vote $\boldsymbol{v} \xleftarrow{\mu} \mathsf{Ch}$. It then computes the aggregated honest voters' vote $\boldsymbol{v}'$ as the sum of all these sampled votes and $\mathsf{V}_{\mathsf{obs}}$'s vote $\boldsymbol{v}_b$ and returns the result $f_{\mathsf{res}}(\boldsymbol{v}_d + \boldsymbol{v}')$.
   Since C is computationally hiding against quantum adversaries and since the published π_{result} is simulated (i.e., it does not depend on the voters' commitments), an admissible adversary cannot distinguish between the published result and a "correctly computed" result corresponding to the honest voters' commitments. Hence, this change can only negligibly affect an admissible adversary's winning probability, i.e.,
   $$|\mathsf{Pr}[\mathbb{E}_5(\mathcal{A}) \mapsto 1] - \mathsf{Pr}[\mathbb{E}_6(\mathcal{A}) \mapsto 1]|$$
   is negligible.
7. $\mathbb{E}_6 \to \mathbb{E}_7$: We define $\mathbb{E}_7$ to be the same as $\mathbb{E}_6$ except for the following change to $\hat{\pi}_{\mathsf{HV}}$ (including $\hat{p}_{\mathsf{V}_{\mathsf{obs}}}$):
   Instead of computing commitments to (shares of) their actual votes, the honest voters publish commitments to the zero-vector.
   Since C is computationally hiding against ppt quantum adversaries and since duplicate ciphertexts are removed by the honest BB, this change can only negligibly affect an admissible adversary's winning probability, i.e.,
   $$|\mathsf{Pr}[\mathbb{E}_6(\mathcal{A}) \mapsto 1] - \mathsf{Pr}[\mathbb{E}_7(\mathcal{A}) \mapsto 1]|$$
   is negligible.
8. $\mathbb{E}_7 \to \mathbb{E}_8$: We define $\mathbb{E}_8$ to be the same as $\mathbb{E}_7$ except for the following change to $\hat{p}_{\mathsf{T}}$:
   Instead of computing the election result as described in the step $\mathbb{E}_5 \to \mathbb{E}_6$, the honest tallier T invokes the ideal voting protocol $\mathsf{P}^{\mathsf{ideal}}(n_v, n^{\mathsf{hon}}_{\mathsf{voters}}, \mathsf{Ch}, f_{\mathsf{res}}, \mu)[\boldsymbol{v}_b]$, where the dishonest voters' votes are extracted from the aggregated sum $\boldsymbol{v}_d$ of all votes cast by dishonest voters. T then returns the result copmuted by the ideal voting protocol. Since the ideal voting protocol $\mathsf{P}^{\mathsf{ideal}}(n_v, n^{\mathsf{hon}}_{\mathsf{voters}}, \mathsf{Ch}, f_{\mathsf{res}}, \mu)[\boldsymbol{v}_b]$ obtains the result just in the same way as T previously did, this change does not affect an adversary's winning probability, i.e.,
   $$|\mathsf{Pr}[\mathbb{E}_7(\mathcal{A}) \mapsto 1] - \mathsf{Pr}[\mathbb{E}_8(\mathcal{A}) \mapsto 1]| = 0.$$

The previous steps have shown that for any adversary $\mathcal{A} \in A_{\mathsf{pub}}$ on $\mathsf{PQKryvos}$, there is an adversary $\mathcal{A}_8$ on $\mathbb{E}_8$ such that the difference

$$|\mathsf{Pr}[\mathbb{E}(\mathcal{A}, \mathsf{P}, \kappa, \mathsf{V}_{\mathsf{obs}}) \mapsto 1] - \mathsf{Pr}[\mathbb{E}_8(\mathcal{A}_8) \mapsto 1]|$$

is negligible. However, note that $\mathcal{A}_8$ does not depend on the honest voters' actual votes except for the publishe election result, which is computed in the same way as in $\mathsf{P}^{\mathsf{ideal}}(n_v, n^{\mathsf{hon}}_{\mathsf{voters}}, \mathsf{Ch}, f_{\mathsf{res}}, \mu)$. Hence, by setting $\mathcal{A}_{\mathsf{ideal}} = \mathcal{A}_8$, we see that

$$\left|\mathsf{Pr}[\mathbb{E}(\mathcal{A}, \mathsf{P}, \kappa, \mathsf{V}_{\mathsf{obs}}) \mapsto 1] - \mathsf{Pr}[\mathbb{E}((\mathcal{A}_{\mathsf{ideal}}, \mathsf{P}^{\mathsf{ideal}}, \kappa, \mathsf{V}_{\mathsf{obs}})) \mapsto 1]\right|$$

is negligible, i.e., $\mathsf{PQKryvos}$ achieves public privacy.

Remark 6. While the proof of Theorem 6 is very similar to the privacy proof in [16], we point out some differences. First, our analysis directly follows the fully game-based approach for analyzing privacy in the KTV framework, which was formalized in [76]. This completely removes the notion of the actual privacy level that even an ideal voting protocol can achieve in a given scenario (based on $\mathsf{Ch}, f_{\mathsf{res}}, \mu, n^{\mathsf{hon}}_{\mathsf{voters}}$), making the privacy of $\mathsf{PQKryvos}$ easier to assess. Moreover, our proof (as in the verifiability proof) also accounts for the (negligible) decryption error that many pq-secure encryption schemes introduce and demonstrates that neither perfect zero-knowledge nor perfectly hiding commitments are required for achieving privacy, both of which were stated as requirements for the privacy analysis in [16].

$\underline{\mathsf{P}^{\mathsf{ideal}}(n_v, n^{\mathsf{hon}}_{\mathsf{voters}}, \mathsf{Ch}, f_{\mathsf{res}}, \mu)[\boldsymbol{v}_b]}$

Parameters:

- Result function f_{res}
- Probability distribution μ over Ch
- Numbers of voters n_v and honest voters $n^{\mathsf{hon}}_{\mathsf{voters}}$
- $I \leftarrow \emptyset$ (initially)

Inputs:

- Choice $\boldsymbol{v}_b$ for the voter under observation $\mathsf{V}_{\mathsf{obs}}$.

$\mathsf{P}^{\mathsf{ideal}}$ uses a scheduler S which forwards messages to and from $\mathcal{A}$ while taking care that the following messages are sent sequentially in the order presented here. It also ensures that the first and last message ($(\mathsf{init}, \mathsf{honest})$ and $\mathsf{compute}$) are sent at most once each:

On $(\mathsf{init}, \mathsf{honest})$ from $\mathcal{A}$ do:

1. $\forall i \in \{1, \ldots, n^{\mathsf{hon}}_{\mathsf{voters}} - 1\}$: store $\boldsymbol{v}_i \xleftarrow{\mu} \mathsf{Ch}$
2. Store $\boldsymbol{v}_{n^{\mathsf{hon}}_{\mathsf{voters}}} \leftarrow \boldsymbol{v}_b$
3. $I \leftarrow I \cup \{1, \ldots, n^{\mathsf{hon}}_{\mathsf{voters}}\}$
4. Return $\mathsf{success}$ (to the adversary $\mathcal{A}$).

On $(\mathsf{setChoice}, i, \boldsymbol{v})$ from $\mathcal{A}$ do:

1. If $i \notin \{n^{\mathsf{hon}}_{\mathsf{voters}} + 1, \ldots, n_v\}$ or $\boldsymbol{v} \notin \mathsf{Ch}$, return $\perp$.
2. Store $\boldsymbol{v}_i \leftarrow \boldsymbol{v}$
3. $I \leftarrow I \cup \{i\}$
4. Return $\mathsf{success}$ (to the adversary $\mathcal{A}$).

On $\mathsf{compute}$ from $\mathcal{A}$ do:

1. Return $\mathsf{res} \leftarrow f_{\mathsf{res}}(\sum_{i \in I} \boldsymbol{v}_i)$ (to the adversary $\mathcal{A}$).

Fig. 4: The ideal voting protocol $\mathsf{P}^{\mathsf{ideal}}$ for choice space Ch, result function f_{res}, number n_v of voters, $n^{\mathsf{hon}}_{\mathsf{voters}} \leq n_v$ of which are honest voters, and vote distribution μ.